TEXAS INSTRUMENTS

TMC57000/TMC57001 /TMS57002

User's Guide

1992

Digital Signal Processing Products

TEXAS INSTRUMENTS

Printed in Japan
'92.08

SCJ1745

Appexdix B Package Dimensions
Package dimensions for the 64–pin SDIP, 80–pin QFP and 68–lead PLCC packages.

Style and Symbol Conventions

This document uses the following conventions.

- Program listings and program examples are shown in a `special typeface` similar to a typewriter's. Here is a sample program listing:

```
NOP1                /DOS 0, D*
MPY     D*+, C*+
```

- In syntax descriptions, any syntax in a **bold typeface** font is required syntax. This includes commas in **bold typeface** which are used to separate parameters. Portions of a syntax that are enclosed in <> describe the type of information that should be entered. Unless syntax in **bold typeface** is enclosed in <>, portions in **bold** should be entered as shown; portions of a syntax that are enclosed in <> describe the type of information that must be entered. Here is an example of a syntax for an instruction description:

  LCAC <constant>

  LCAC is the instruction description mnemonic. When you use **LCAC**, **LCAC** must be inputted exactly as shown and the parameter must be an actual constant.

- Braces ({ and }) indicate a list in which one item must be chosen. The symbol / separates items within the list. Here's an example of a list used within an instruction description:

  OR {D*/D*+}, A

  This provides two choices: `D*` or `D*+`. Therefore, valid syntax when using the OR instruction is one of the following two examples:

  OR D*, A
  or
  OR D*+, A

- When referring to the contents of a location in memory, the name of memory is followed by the actual address location in parentheses:

  PMEM(0)
  DMEM(32)
  CMEM(255)

The above refers to the contents of PMEM(Program Memory) at address 0, the contents of DMEM(Data Memory) at address 32, and the contents of CMEM(Coefficient Memory) at address 255.

Chapter 1

Introduction

The 5700x series of digital signal processors is a high performance family of products designed especially for audio applications. Texas Instruments designed the TMC57000 and TMC57001 to efficiently implement linear phase FIR filters (up to 500 taps). For the implementation of audio applications that require handling delayed data, the architecture of the TMC57000 and TMC57001 was expanded to include an external RAM interface.

Plans for expansion of the 5700x series include more powerful digital signal processors designed to meet the increasing market demand for DSP solutions to audio applications.

This document discusses the three digital signal processors in the 5700x series, the TMC57000, TMC57001 and TMS57002.

Topic **Page**

1.1	**5700x Key Features**	**1-2**
1.2	**Typical Applications**	**1-3**

1.1 5700x Key Features

Some of the key features of the 5700x devices are listed below. Features which pertain to a particular device are noted by enclosing the device name within parentheses.

- ❑ Instruction cycle timing: 80 ns
- ❑ Parallel instruction execution
- ❑ Program Memory: 256 x 24–bit SRAM
- ❑ Coefficient Memory: 256 x 32–bit SRAM
- ❑ Audio Data Memory:
  - 256 x 24–bit x 2 DRAMs (TMC57000/1)
  - 256 x 24–bit and 32 x 24–bit DRAMs (TMS57002)
- ❑ External Memory Interface (TMS57002):
  - 256K – 8 M bit external RAM
  - 4 or 8–bit I/O data length
  - 16 or 24–bit audio data word length
  - DRAM or SRAM
  - external memory used as ring buffer for audio data
- ❑ 32–bit ALU
- ❑ 25 x 32–bit Multiplier Accumulator (MAC)
- ❑ 4 data buses:
  - two 24–bit buses
  - two 32–bit buses
- ❑ 2 serial input ports for audio data
- ❑ 2 serial output ports for audio data
- ❑ 8–bit parallel port for host interface
- ❑ Parallel execution of data transfers and calculations
- ❑ Cascade capability
- ❑ Up to 50 stage second order IIR filter
- ❑ Up to 500 taps linear phase FIR filter (TMC57000/1)
- ❑ Up to 1500 taps linear phase FIR filter by cascading 3 devices (TMC57001)

- ❑ Device packaging:
  - 64–pin plastic shrink dual–in–line package (TMC57000/1)
  - 80–pin plastic quad flat pack package (TMC57001,TMS57002)
  - 68–lead plastic leaded chip carrier package (TMS57002)

1.2 Typical Applications

The 5700x devices offers a high performance solution for a variety of applications, in particular audio applications. Following are a list of 5700x family applications:

- ❑ Digital Filters
- ❑ Surround Processing
- ❑ Dynamic Range Control
- ❑ Equalizer
- ❑ Sound Effector

Chapter 2

Pinouts and Signal Descriptions

The 5700x series digital signal processors are available in one or two package types. The TMC57000 is available in a 64–pin plastic shrink dual–in–line package (SDIP). The TMC57001 is available in a 64–pin plastic SDIP or an 80–pin plastic quad flat pack (QFP) package. The TMS57002 is available in an 80–pin plastic QFP package or a plastic 68–lead chip carrier (PLCC) package.

This chapter presents a simple layout of 64–pin SDIP, 80–pin QFP and 68–lead PLCC packages, with pin and signal names, and a table of signal definitions for each of the devices.

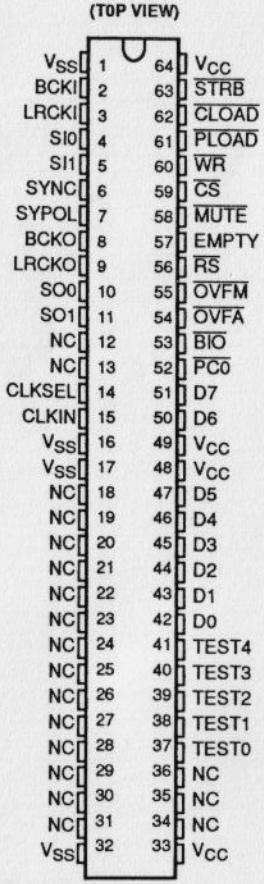
Topic **Page**

2.1	Pin Layout	2-2
2.2	Signal Descriptions	2-5

2.1 Pin Layout

Figure 2–1 shows pinouts of the 64–pin plastic SDIP package for the TMC57000 and TMC57001.

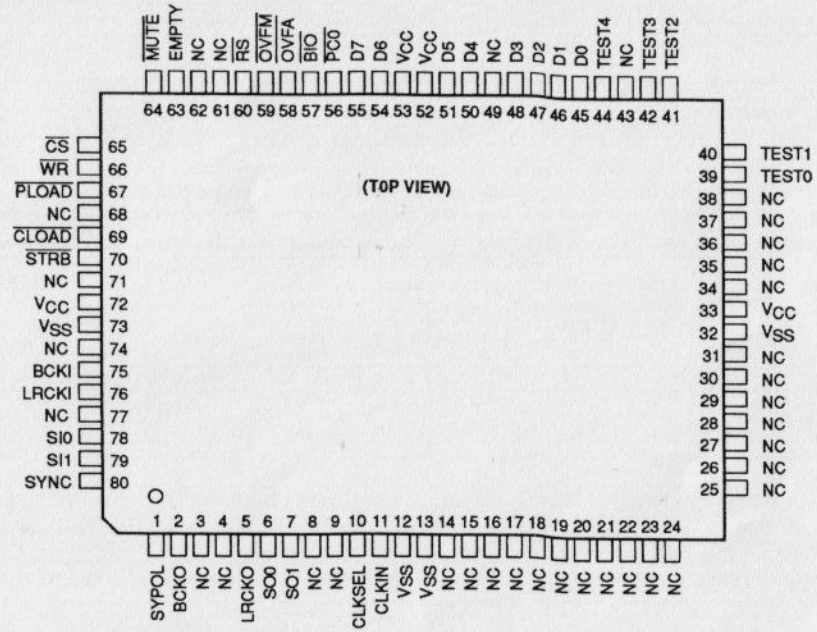
Figure 2–1. Signal Assignments for TMC57000 and TMC57001 64–pin plastic SDIP



Note: NC=No connect. (These pins are reserved.)

Figure 2–2 shows pinouts of the 80–pin plastic QFP package for the TMC57001.

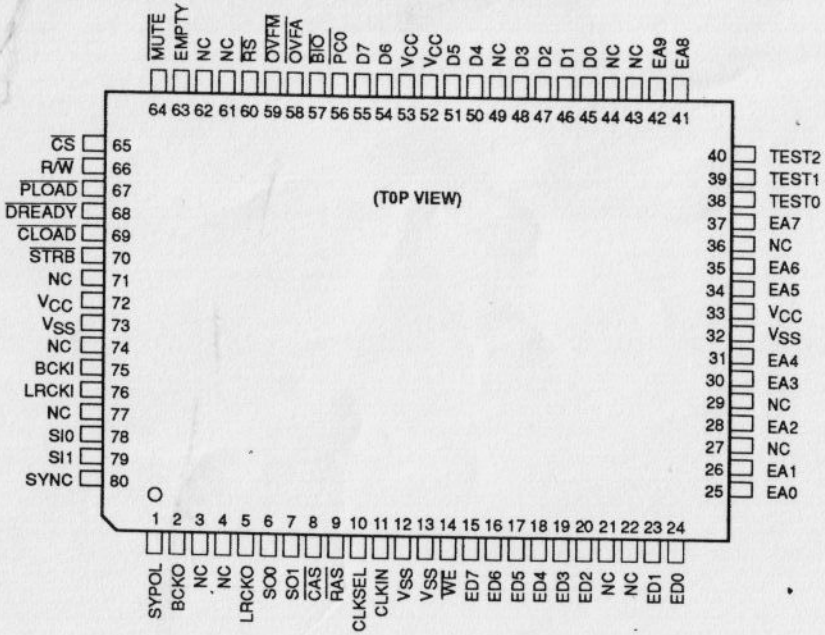
Figure 2–2. Signal Assignments for TMC57001 80–pin plastic QFP



Note: NC=No connect. (These pins are reserved.)

Figure 2–3 shows pinouts of the 80–pin plastic QFP package for the TMS57002.

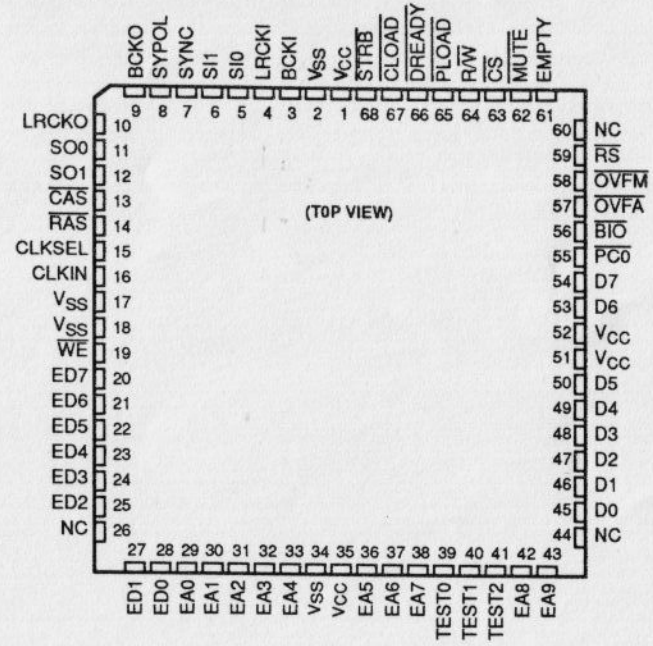
Figure 2–3. Signal Assignments for TMS57002 80–pin plastic QFP



Note: NC=No connect. (These pins are reserved.)

Figure 2–4 shows pinouts of the 68–lead PLCC for the TMS57002.

Figure 2–4. Signal Assignments for TMS57002 68–lead PLCC



Note: NC=No connect. (These pins are reserved.)

2.2 Signal Descriptions

The signals for the 5700x are described in this section. Table 2–1 and Table 2–2 list each signal, its pin location, function, and operating mode(s), i.e., input (I), output (O), high–impedance (Z), open drain (OD) or supply (S) state. The signals are grouped according to their functional purpose.

Table 2–1. TMC57000/1 Signal Descriptions for 64–pin SDIP and 80–pin QFP

Signal	Pin		State	Description
	SDIP	QFP		
Address/Data Buses				
D7 (MSB) D6 D5 D4 D3 D2 D1 D0 (LSB)	51 50 47 46 45 44 43 42	55 54 51 50 48 47 46 45	I	Parallel data bus D7(MSB) through D0(LSB) for data transfers from the host controller to the TMC57000 or TMC57001.
Host Interface Control Signals				
$\overline{\text{CLOAD}}$	62	69	I	Host interface mode select input. Normally high, unless low level asserted for CMEM download or update operations during $\overline{\text{CS}}$ and $\overline{\text{WR}}$ are low.
$\overline{\text{CS}}$	59	65	I	Chip select input. A low on this input enables host interface operations.
$\overline{\text{PLOAD}}$	61	67	I	Host interface mode select input. A low on this input enables host interface download operations (ST0/ST1/PMEM download and CMEM download).
$\overline{\text{STRB}}$	63	70	I	Strobe input. A low pulse on this input latches data on the D7 through D0 pins into the HOST registers (four 8–bit host interface buffers).
$\overline{\text{WR}}$	60	66	I	Write enable for host interface operations. $\overline{\text{WR}}$ may be kept low at all times because read operations are available only for device test.
Serial Port Signals				
BCKI	2	75	I	Receive clock input. External clock signal for clocking data from the SI0/SI1 (data input) pins into the SHIFTIN_0/SHIFTIN_1 (serial port input shift registers).
BCKO	8	2	I	Transmit clock input. External clock signal for clocking data from SHIFTOUT_0/SHIFTOUT_1 (serial port output shift registers) to the SO0/SO1 (data output) pins.
LRCKI	3	76	I	Left/right channel receive clock input. During the first half of the LRCKI period, one word of left channel data is received and during the second half of the LRCKI period, one word of right channel data is received. LRCKI and BCKI must be synchronous.

Table 2–1. TMC57000/1 Signal Descriptions for 64–pin SDIP and 80–pin QFP (Continued)

Signal	Pin		State	Description
	SDIP	QFP		
Serial Port Signals (Continued)				
LRCKO	9	5	I	Left/right channel transmit clock input. During the first half of the LRCKO period, one word of left channel data is transmitted and during the second half of the LRCKO period, one word of right channel data is transmitted. LRCKO and BCKO must be synchronous.
$\overline{\text{MUTE}}$	58	64	I	Mute control input. A low on this input sets the output at SO0/SO1 to zero. This operation is synchronous with LRCKO.
SI0 SI1	4 5	78 79	I I	Serial port receive inputs. Serial data is received in SHIFTIN_0/SHIFTIN_1 (serial port input shift registers) via the SI0/SI1 pins.
SO0 SO1	10 11	6 7	O/Z O/Z	Serial port transmit outputs. Serial data is transmitted from SHIFTOUT_0/SHIFTOUT_1 (serial port output shift registers) via the SO0/SO1 pins.
Miscellaneous Signals				
$\overline{\text{BIO}}$	53	57	I	Branch control input. If low when using the BIOZ instruction, the conditional instruction is executed.
CLKIN	15	11	I	Master clock input.
CLKSEL	14	10	I	Master clock mode select. If CLKSEL=0, the internal machine cycle is half the clock rate at CLKIN. If CLKSEL=1, the internal machine cycle and CLKIN clock rate are same.
EMPTY	57	63	O	Empty flag output. When the 16 UPDATE registers of the host interface are empty, EMPTY is set high. The UPDATE registers hold coefficient values for CMEM update. After CMEM has been updated with the UPDATE registers contents, EMPTY is set high.
$\overline{\text{OVFA}}$	54	58	OD	ALU overflow flag output. The value of $\overline{\text{AOV}}$ (ALU overflow flag in ST1) is inverted at the $\overline{\text{OVFA}}$ pin. This pin is an open drain output.
$\overline{\text{OVFM}}$	55	59	OD	MAC overflow flag output. The value of $\overline{\text{MOV}}$ (MAC overflow flag in ST1) is inverted at the $\overline{\text{OVFM}}$ pin. This pin is an open drain output.
$\overline{\text{PC0}}$	52	56	O	PC zero detect output. $\overline{\text{PC0}}$ is set to active low when PC=0.
$\overline{\text{RS}}$	56	60	I	Hardware reset. Causes the device to terminate execution and forces the program counter to zero. Affects various registers and status bits.
SYNC	6	80	I	Program synchronization input. Synchronization is detected on the falling edge of the internal master clock. When CLKSEL=0, synchronization is detected at either the first or second rising edge of CLKIN because the internal master clock is divide–by–two clock at CLKIN. When CLKSEL=1, synchronization is detected on the falling edge of CLKIN.

Table 2–1. TMC57000/1 Signal Descriptions for 64–pin SDIP and 80–pin QFP (Continued)

Signal	Pin		State	Description
	SDIP	QFP		
Miscellaneous Signals (Continued)				
SYPOL	7	1	I	SYNC pin trigger edge select. If SYNC=0, falling edge trigger. If SYNC=1, rising edge trigger.
Test/Supply/Reserved Pins				
V_{CC}	33 48 49 64	33 52 53 72	S	Four +5 volt power supply pins, tied together externally.
V_{SS}	1 16 17 32	73 12 13 32	S	Four ground pins, tied together externally.
TEST0 TEST1 TEST2 TEST3 TEST4	37 38 39 40 41	39 40 41 42 44	I	Test pins with internal pull–ups. These pins are reserved for device test. These pins should be left unconnected.
NC	12 13 18 19 20 21 22 23 24 25 26 27 28 29 30 31 34 35 36	3 4 8 9 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 34 35 36 37 38 43 49 61 62 68 71 74 77		Reserved pins. These pins should be left unconnected.

Table 2–2. TMS57002 Signal Descriptions for 68–lead PLCC and 80–pin QFP

Signal	Pin		State	Description
	QFP	PLCC		
Address/Data Buses				
D7 (MSB) D6 D5 D4 D3 D2 D1 D0 (LSB)	55 54 51 50 48 47 46 45	54 53 50 49 48 47 46 45	I/O	Parallel data bus D7(MSB) through D0(LSB) for data transfers between the host controller and TMS57002.
EA9 (MSB) EA8 EA7 EA6 EA5 EA4 EA3 EA2 EA1 EA0 (LSB)	42 41 37 35 34 31 30 28 26 25	43 42 38 37 36 33 32 31 30 29	O	Parallel address bus EA9 (MSB) through EA0 (LSB) for external RAM. The 10–bit row and column addresses are output in a time–shared fashion to these pins.
ED7 (MSB) ED6 ED5 ED4 ED3 ED2 ED1 ED0 (LSB)	15 16 17 18 19 20 23 24	20 21 22 23 24 25 27 28	I/O	Parallel data bus ED7(MSB) through ED0(LSB) for external RAM. ED3 through ED0 are not used if SEL is 0. These pins are with internal pull–ups.
External Memory Signals				
$\overline{CAS}$	8	13	O	Column address strobe for external RAM.
$\overline{RAS}$	9	14	O	Row address strobe for external RAM.
$\overline{WE}$	14	19	O	Write enable for external RAM.
Host Interface Control Signals				
$\overline{CLOAD}$	69	67	I	Host interface mode select input. Normally high, unless low level asserted for CMEM download or update operations during $\overline{CS}$ and $\overline{WR}$ are low.
$\overline{CS}$	65	63	I	Chip select input. A low on this input enables host interface operations.
$\overline{DREADY}$	68	66	O	Data ready signal to host controller. Set low upon execution of the LPC the LPD instruction to signal that a word of CMEM or DMEM data has been loaded into the HOST registers during $\overline{CS}$ is low and R/$\overline{W}$ is high.
$\overline{PLOAD}$	67	65	I	Host interface mode select input. A low on this input enables host interface download operations (ST0/ST1/PMEM download and CMEM download).

Table 2–2. TMS57002 Signal Descriptions for 68–lead PLCC and 80–pin QFP (Continued)

Signal	Pin		State	Description
	QFP	PLCC		
Host Interface Control Signals (Continued)				
$\overline{STRB}$	70	68	I	A low pulse on this input reads/writes host interface data via D7 through D0 if $\overline{CS}$ is low.
R/$\overline{W}$	66	64	I	Read/Write control signal for host interface operations. A low level on this pin enables write operation to the TMS57002 during $\overline{CS}$ is low. A high level on this pin enables read operation from the TMS57002 during $\overline{CS}$ is low.
Serial Port Signals				
BCKI	75	3	I	Receive clock input. External clock signal for clocking data from the SI0/SI1 (data input) pins into the SHIFTIN_0/SHIFTIN_1 (serial port input shift registers).
BCKO	2	9	I	Transmit clock input. External clock signal for clocking data from SHIFTOUT_0/SHIFTOUT_1 (serial port output shift registers) to the SO0/SO1 (data output) pins.
LRCKI	76	4	I	Left/right channel receive clock input. During the first half of the LRCKI period, one word of left channel data is received and during the second half of the LRCKI period, one word of right channel data is received. LRCKI and BCKI must be synchronous.
LRCKO	5	10	I	Left/right channel transmit clock input. During the first half of the LRCKO period, one word of left channel data is transmitted and during the second half of the LRCKO period, one word of right channel data is transmitted. LRCKO and BCKO must be synchronous.
SI0 SI1	78 79	5 6	I I	Serial port receive inputs. Serial data is received in SHIFTIN_0/SHIFTIN_1 (serial port input registers) via the SI0/SI1 pins.
SO0 SO1	6 7	11 12	O/Z O/Z	Serial port transmit outputs. Serial data is transmitted from SHIFTOUT_0/SHIFTOUT_1 (serial port output shift registers) via the SO0/SO1 pins.
Miscellaneous Signals				
$\overline{BIO}$	57	56	I	Branch control input. If low when using the BIOZ instruction, the conditional instruction is executed.
CLKSEL	10	15	I	Master clock mode select. If CLKSEL=0, the internal machine cycle is half the clock rate at CLKIN. If CLKSEL=1, the internal machine cycle and CLKIN clock rate are same.
CLKIN	11	16	I	Master clock input.

Table 2–2. TMS57002 Signal Descriptions for 68–lead PLCC and 80–pin QFP (Continued)

Signal	Pin		State	Description
	QFP	PLCC		
Miscellaneous Signals (Continued)				
EMPTY	63	61	O	Empty flag output. When the 16 UPDATE registers of the host interface are empty, EMPTY is set high. The UPDATE registers hold coefficient values for CMEM update. After CMEM has been updated with the UPDATE registers contents, EMPTY is set high.
$\overline{MUTE}$	64	62	I	Mute control input. A low on this input sets the output at SO0/SO1 to zero. This operation is synchronous with LRCKO.
$\overline{OVFA}$	58	57	OD	ALU overflow flag output. The value of AOV (ALU overflow flag in ST1) is inverted at the $\overline{OVFA}$ pin. This pin is an open drain output.
$\overline{OVFM}$	59	58	OD	MAC overflow flag output. The value of MOV (MAC overflow flag in ST1) is inverted at the $\overline{OVFM}$ pin. This pin is an open drain output.
$\overline{PC0}$	56	55	O	PC zero detect output. $\overline{PC0}$ is set to active low when PC=0.
$\overline{RS}$	60	59	I	Hardware reset. Causes the device to terminate execution and forces the program counter to zero. Affects various registers and status bits.
SYNC	80	7	I	Program synchronization input. Synchronization is detected on the falling edge of the internal master clock. When CLKSEL=0, synchronization is detected at either the first or second rising edge of CLKIN because the internal master clock is divide–by–two clock at CLKIN. When CLKSEL=1, synchronization is detected on the falling edge of CLKIN.
SYPOL	1	8	I	SYNC pin trigger edge select. If sync=0, falling edge trigger. If SYNC=1, rising edge trigger.
Test/Supply/Reserved Pins				
V_{CC}	33 52 53 72	35 51 52 1	S	Four +5 volt power supply pins, tied together externally.
V_{SS}	12 13 32 73	17 18 34 2	S	Four ground pins, tied together externally.
TEST0 TEST1 TEST2	38 39 40	39 40 41	I	Test pins with internal pull–ups. These pins are reserved for device test. These pins should be left unconnected.

Table 2–2. TMS57002 Signal Descriptions for 68–lead PLCC and 80–pin QFP (Continued)

Signal	Pin		State	Description
	QFP	PLCC		
Test/Supply/Reserved Pins				
NC	3 4 21 22 27 29 36 43 44 49 61 62 71 74 77	26 44 60		Reserved pins. These pins should be left unconnected.

Chapter 3

Architecture

The architectural design of the 5700x Series is discussed in this chapter. The basic architecture of the TMC57000, TMC57001 and TMS57002 is the same. Any differences in architecture among the 5700x devices will be noted in the explanation.

Major topics discussed in this chapter are listed below.

Topic		Page
3.1	Architectural Overview	3-2
3.2	Internal Hardware	3-5
3.3	Internal Memory Organization	3-9
3.4	Arithmetic Logic Unit (ALU)	3-16
3.5	Multiplier Accumulator (MAC)	3-18
3.6	System Control	3-24
3.7	Host Interface	3-37
3.8	Serial Port	3-47
3.9	External RAM Interface (TMS57002 only)	3-63

3.1 Architectural Overview

The 5700x family of devices utilizes a Harvard Architecture with four data buses for speed and flexibility. This architecture makes possible the parallel execution of two instructions which reduces the number of program lines. In addition, execution time is reduced by this architecture which allows for the simultaneous access of data in two internal RAMs for arithmetic, logical and multiplicative operations. Internal memory consists of program RAM, coefficient RAM, and two banks of data RAM. Refer to Figure 3–1 and Figure 3–2 for the architecture of the TMC57000, TMC57001 and TMS57002.

With the TMC57000 and TMC57001, a linear phase FIR filter with up to 500 taps can be implemented. With the TMC57001, by cascading 3 devices, a 1500 taps linear phase filter can be realized. All 5700x devices can be cascaded. The 5700x devices can also be used for second order IIR filter calculations of up to 50 stages.

The 5700x Calculation Unit contains both a 32–bit ALU and a 24 x 32–bit Multiplier Accumulator (MAC). The ALU performs twos–complement arithmetic and logical functions. The Multiplier Accumulator performs twos–complement fixed–point multiplication with accumulation. Both signed and unsigned multiplication are supported. A fast and efficient implementation of a linear phase FIR filter is possible due to these two separate calculation units and also due to the simultaneous access of data in two RAMs.

The serial I/O port contains two input ports and two output ports. User–programmable options are available which control the input and output data format. Options include the data word length (16 or 24–bit), the order of bits when a data word is transmitted or received (MSB first or LSB first) and the data format (Standard or I^2S Format).

The host interface is an 8–bit parallel interface between the 5700x device and the host controller. The host interface downloads the contents of the status registers, program memory and coefficient memory at processor initialization. In addition, (with the TMS57002 only) data memory and coefficient memory values can be outputted to the host interface.

The TMS57002 includes an external RAM interface which allows for the implementation of audio application that require handling delayed data. The following user–programmable options are available:

- ❑ 256K to 8M–bit external RAM
- ❑ DRAM or SRAM compatibility
- ❑ 4 or 8–bit I/O port width
- ❑ 16 or 24–bit data word length

The 5700x devices permit parallel execution of primary and secondary instructions. The two lines below,

```
NOP1              /DIS 0, D*
MPY    D*+, C*+
```

can be reduced to a single line as follows:

```
MPY    D*+, C*+ /DIS 0
```

In all cases, as in the example above, where one instruction requires read access to memory and the other instruction requires write access to memory, the instruction with write access takes precedence.

Figure 3–1. Architecture of the TMC57000 and TMC57001

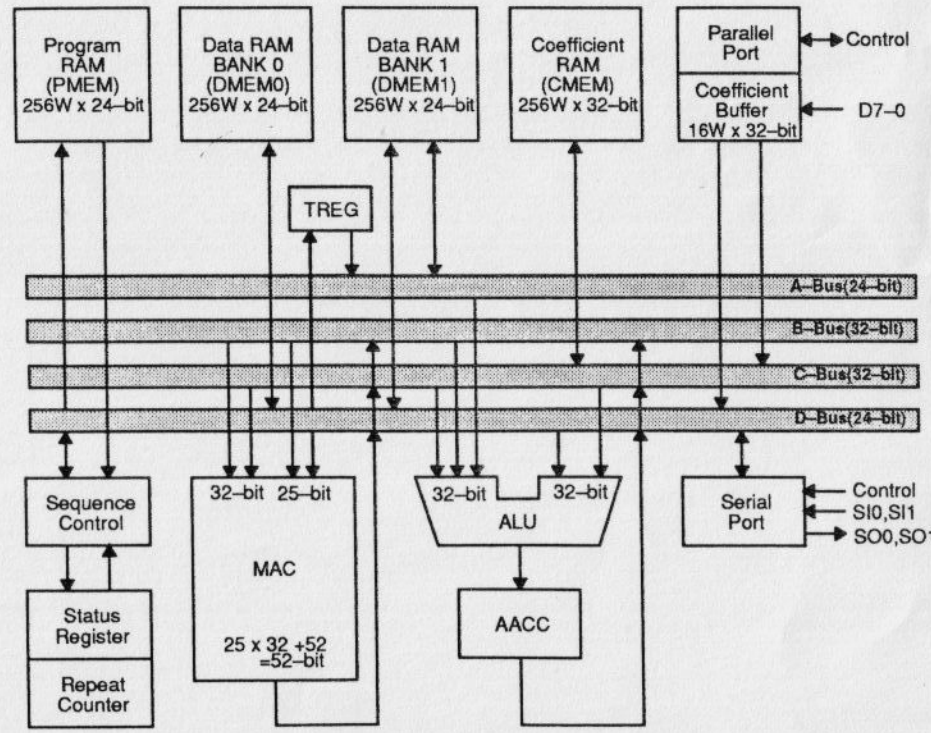
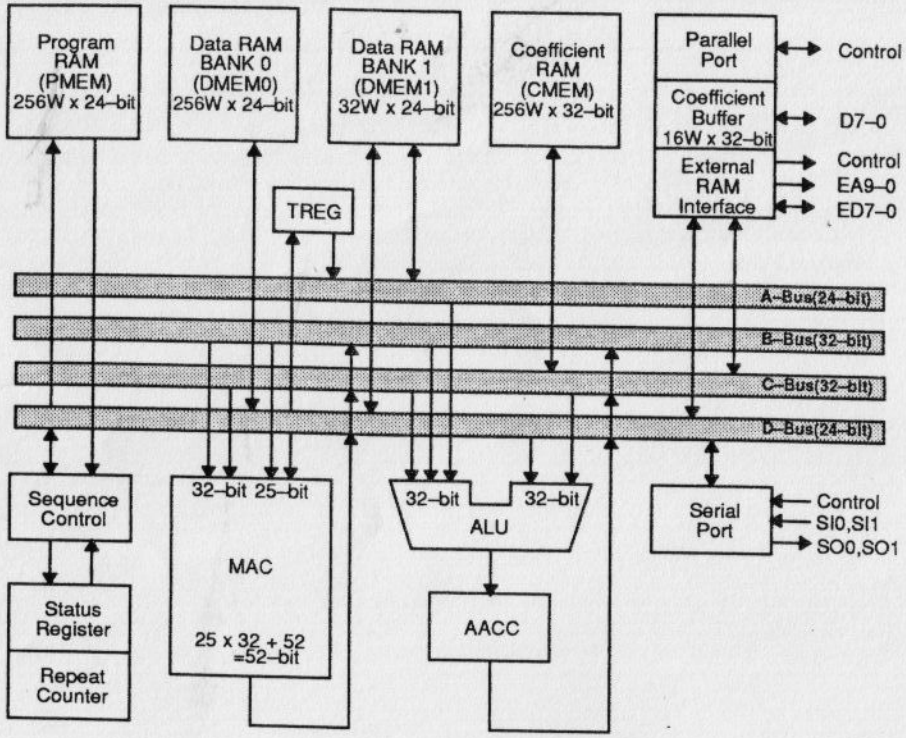


Figure 3–2. Architecture of the TMS57002



3.2 Internal Hardware

This chapter includes a summary of the 5700x devices' internal hardware and an explanation of the data format.

3.2.1 Internal Hardware Summary

Below is a summary of the 5700x series internal hardware. This summary, which includes the internal processing elements, registers and buses, is alphabetized within each functional grouping. All of the symbols used in this summary will be used consistently throughout this document.

Table 3–1. Internal Hardware Summary

Unit	Symbol	Function
Buses		
Local Data Bus	A–Bus	A 24–bit bus used to route DMEM1 (Data Memory Bank1) data to the ALU or from the TREG.
ALU and MAC Output Bus	B–Bus	A 32–bit bus used to route the ALU and MAC (Multiplier Accumulator) output.
Coefficient Memory Bus	C–Bus	A 32–bit bus used to route CMEM (Coefficient Memory) data.
Data Memory Bus	D–Bus	A 24–bit bus used to route DMEM (Data Memory) data and serial input/output.
Instruction Bus	IR–Bus	A 24–bit program memory bus used to route instruction code data.
Calculation Units		
Arithmetic Logic Unit	ALU	A 32–bit twos–complement arithmetic logic unit with two 24 or 32–bit inputs and a 32–bit accumulator. Performs fixed point arithmetic.
Multiplier Accumulator	MAC	A 25 x 32–bit multiplier with a 52–bit accumulator. Performs fixed point calculations.
Memory		
Coefficient Memory	CMEM	256 x 32–bit word static RAM.
Data Memory	DMEM (DMEM0 and DMEM1)	Two dynamic RAMs for storing sampled data: TMC57000/1: DMEM0: 256 x 24–bit DRAM DMEM1: 256 x 24–bit DRAM TMS57002: DMEM0: 256 x 24–bit DRAM DMEM1: 32 x 24–bit DRAM
Program Memory	PMEM	256 x 24–bit word static RAM for program memory.
Registers		
ALU Accumulator	AACC	A 32–bit accumulator for storage of ALU output.
Base Address Registers	BA0 BA1	An 8–bit base address register for DMEM0. An 8–bit base address register for DMEM1.
C–Register for a 32–bit MAC Input	CREG	A 32–bit register that stores a 32–bit input to the MAC from CMEM, the ALU or the MAC.

Table 3–1. Internal Hardware Summary (Continued)

Unit	Symbol	Function
Registers (Continued)		
Coefficient Address Register	CA	An 8–bit address for indirect addressing of CMEM.
D–Register for a 24/25–bit MAC Input	DREG	A 25–bit register that stores a 24/25–bit input to the MAC from DMEM, the ALU or the MAC.
Empty Flag Registers	EmptyX $0 \leqq X < 16$	Sixteen 1–bit flags, each assigned to one of the corresponding sixteen Update Registers. When set, signals that the Update Register value has been used for CMEM update. When all Empty Flag Registers are set, the EMPTY pin is set signalling CMEM update is complete.
Index Register	ID	An 8–bit offset value from the base address for indirect addressing of DMEM.
Update Registers	UpdateX $0 \leqq X < 16$	Sixteen 32–bit buffers that store CMEM update values.
Instruction Register	IR	A 24–bit register that stores instruction code.
MAC Accumulator	MACC	52–bit accumulator for storage of Multiplier Accumulator (MAC) output.
Program Counter	PC	An 8–bit program counter used to address PMEM. The PC always contains the address of the next instruction to be executed.
Host Interface Registers	HOST0 HOST1 HOST2 HOST3	Four 8–bit registers which hold one word (1 word = 24 or 32–bit) of data during host interface operations.
Repeat Counter	RPTC	An 8–bit counter for repeated instruction execution control.
Serial Input Registers for Left Channel Data	SI0_L SI1_L	24–bit registers that receive one word of left channel serial input data.
Serial Input Registers for Right Channel Data	SI0_R SI1_R	24–bit registers that receive one word of right channel serial input data.
Serial Output Registers for Left Channel Data	SO0_L SO1_L	24–bit registers that transmit one word of left channel serial output data.
Serial Output Registers for Right Channel Data	SO0_R SO1_R	24–bit registers that transmit one word of right channel serial output data.
Starting Address Register	SA	8–bit register that holds the CMEM address at which CMEM update will begin.
Status Registers	ST0 ST1	Two 24–bit registers that contain status and control bits.
Temporary Register	TREG	Temporary data storage (24–bit) for data transfer between DMEM0 and DMEM1.

Table 3–1. Internal Hardware Summary (Continued)

Unit	Symbol	Function
External RAM Interface Registers (TMS57002 only)		
External RAM Interface Base Address	XBA	19–bit base address register for external RAM.
External RAM Interface Offset Address	XOA	19–bit offset address register for external RAM.
External RAM Interface Read Data Buffer	XRD	24–bit buffer for one word of external RAM data that will be stored in DMEM.
External RAM Interface Write Data Buffer	XWR	24–bit buffer for one word of DMEM data that will be written to external RAM.
Others		
ALU Input Shifters	Input Shifter A/B	Right shifters (–1 or 0–bit) that shift ALU input from A–Bus and/or D–Bus.
ALU Output Shifter	–	A left shifter (0 or 7–bit) that shifts the ALU output.
MAC Input Shifter	–	A left shifter (0 or 1–bit) that shifts the MAC input from the B–Bus.
MAC Output Shifter	–	A shifter (–8, 0, +2 or +4–bit) that shifts the MAC output.
MAC Output Rounder	–	An adder that rounds the MAC output to a 16, 20, 24, 32 or 48–bit value.
Accumulation Shifter	–	A shifter (–16, 0, +2 or +4–bit) that shifts the MACC output which is then inputted to the MAC for accumulation.

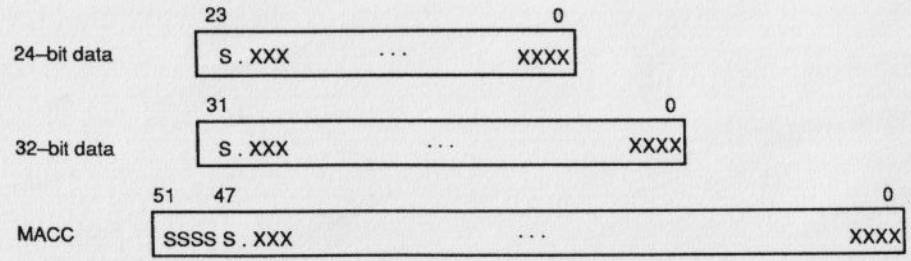
3.2.2 Data Formats

The 5700x supports two data formats: a fractional format and an unsigned format. The majority of the data is in the fractional format. Data in the fractional format is represented in twos–complement notation. The range of the value is:

$$-1 \leq x < +1$$

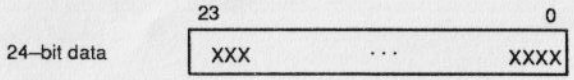
The location of the sign bit and the decimal point for 24–bit data, 32–bit data and the 52–bit MACC is illustrated in Figure 3–3 below.

Figure 3–3. Fractional Data Format



The unsigned format is available only for 24–bit data memory values when using the MACU and MPYU instructions. There is no sign bit or decimal point as illustrated below in Figure 3–4.

Figure 3–4. Unsigned Data Format



3.3 Internal Memory Organization

This chapter includes the following descriptions:

- Program Memory and Addressing
- Coefficient Memory and Addressing
- Data Memory and Addressing

3.3.1 Program Memory and Addressing

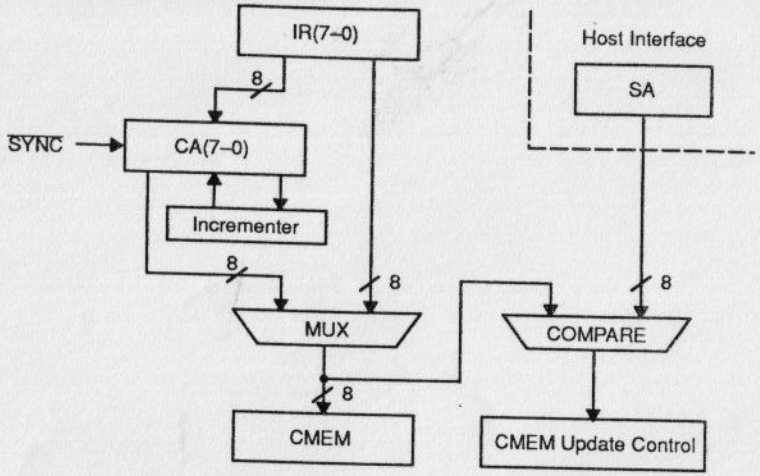
Program Memory (PMEM) consists of a 256 x 24–bit SRAM. PMEM is addressed with an 8–bit program counter (PC) which is incremented by each instruction execution except branch executions. The PC is set to zero at synchronization and hardware reset. In an instruction repeat loop, the PC is not incremented until the repeat counter(RPTC) reaches zero.

The PC will also be reset to zero after reaching 255. In cases where the PC will be reset to zero before synchronization, use the IDLE instruction to disable the incrementation of the PC until synchronization clears the PC to zero.

3.3.2 Coefficient Memory and Addressing

Coefficient Memory (CMEM) consists of a 256 x 32–bit SRAM. The CMEM addressing unit is illustrated in Figure 3–5. The CMEM addressing unit calculates both direct and indirect addresses for CMEM access. If the host interface is in CMEM update mode, the address calculated will be compared to the value of the SA register. If the values are equal, this signals the host interface to begin updating the data in CMEM. Please refer to Section 3.7 for more details on the host interface.

Figure 3–5. Coefficient Memory Addressing Unit



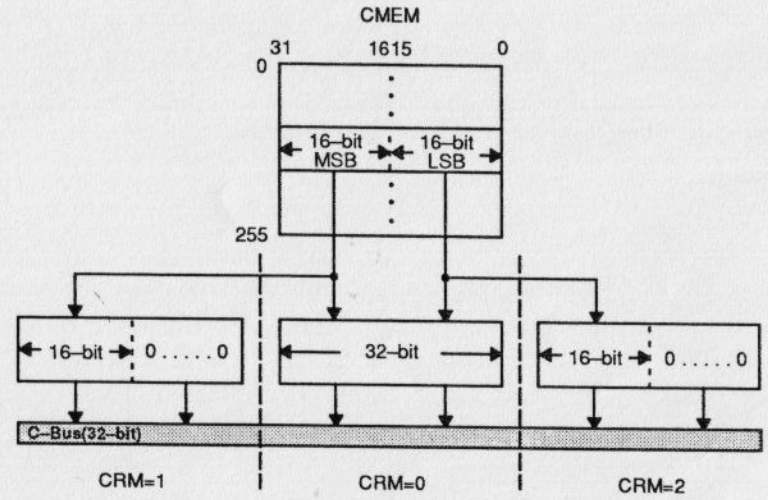
For direct addressing, IR (bits 7 to 0) contains the access address into CMEM. For indirect addressing, the CA register contains the access address into CMEM. For indirect addressing with post–increment, the CA register is incremented by one after the CMEM access. The CA register is reset to zero at synchronization and also when downloading CMEM data (refer to Section 3.7.2). The CA register can also be set directly with the following instructions:

- ❑ LCAA: Load the CA with AACC
- ❑ LCAC: Load the CA conditionally
- ❑ LCAK: Load the CA with an immediate value

Please refer to these instruction descriptions for more details.

When reading a 32–bit value from CMEM, the format of one word of data is controlled by the control bit CRM. Please refer to Figure 3–6. If CRM=0, a 32–bit CMEM value is outputted to the C–Bus. If CRM=1, the 16 high order bits of the CMEM word are outputted and the low order bits of the output are filled with zeros. If CRM=2, the 16 low order bits of the CMEM word are outputted to the 16 high order bits of the C–Bus and the 16 low order bits of the output are filled with zeros.

Figure 3–6. CMEM Word Format for Read Access



3.3.3 Data Memory and Addressing

All 5700x devices contain two data memory banks, DMEM0 and DMEM1. For all 5700x devices, DMEM0 is a 256 x 24–bit DRAM. In the TMC57000 and TMC57001, DMEM1 is also a 256 x 24–bit DRAM, whereas in the TMS57002, DMEM1 is a 32 x 24–bit DRAM.

Although the basic data memory architecture is the same for all 5700x devices, there are slight differences in the data memory architecture. The addressing unit of the TMC57000 and TMS57002(CAS=0) is illustrated in Figure 3–7 and the addressing unit of the TMC57001 and TMS57002(CAS=1) is illustrated in Figure 3–8.

The addressing scheme of the TMS57002 is controlled by the control bit CAS. If CAS=0, the addressing unit of the TMS57002 is identical to that of the TMC57000. If CAS=1, the addressing unit of the TMS57002 is identical to that of the TMC57001.

Both DMEM0 and DMEM1 are implemented as circular buffers with a length of 256 (DMEM1 of TMS57002 is of length 32). DMEM0 and DMEM1 can also be used to implement multiple delay buffers. The circular buffer implementation of internal memory is discussed in detail in Section 5.3.

The base address of DMEM0 is BA0 and the base address of DMEM1 is BA1. If INCS=0, at synchronization, BA0 is decremented by one and BA1 is incremented by one. If INCS=1, this incrementing and decrementing function is disabled. BA0 can be set with an immediate value using the LBRK instruction. BA1 can only be cleared by reset.

DMEM1, BA1 and TREG have been provided especially for the implementation of linear phase FIR filters. With 5700x devices, linear phase FIR filters can be implemented in approximately half the processing time of conventional DSP.

For linear phase FIR filters it is important to access two data samples simultaneously for pre–addition. With the ADDS, AMPY and AMAC instructions, DMEM0 and DMEM1 are accessed simultaneously. BA0 is used as the base address for DMEM0, and BA1 is used as the base address for DMEM1. TREG is used to transfer a data value from DMEM0 to DMEM1 with the LD0T and STD1 instructions. Refer to Section 5.4 for more on linear phase FIR filters.

With the ADDS, AMPY and AMAC instructions, DMEM0 and DMEM1 are accessed simultaneously, but with all other instructions, only one memory bank can be accessed at a time. The DBP control bit controls which memory bank is selected. If DBP=0, DMEM0 is selected. If DBP=1, DMEM1 is selected. Only with the above instructions is DBP ignored.

With the TMC57000, applications requiring up to 512 delayed data samples can be implemented. DMEM0 and DMEM1 have been implemented as memory pages (1 page=256 words). DBP points to either DMEM0 or DMEM1 and BA0 is used to calculate the memory address. Both DMEM0 and DMEM1 are implemented as circular buffers of length 256. Multiple delay buffers can also be implemented. Refer to Section 5.3 for more on circular buffer implementation.

With the TMC57001 and TMS57002(CAS=1), DMEM0 is also implemented as a circular buffer, but DMEM1 is only implemented as a circular buffer when using the ADDS, AMAC and ADDS instructions for linear phase FIR filters. Up to three TMC57001 devices can be cascaded for implementing up to a 1500–tap linear phase FIR filter. Indirect addressing of DMEM1 using the BA1 base address is available which makes this possible. Refer to Section 5.5 and Section 5.6 for applications of cascaded devices.

Figure 3–7. Data Memory Addressing Unit of the TMC57000 and TMS57002 (CAS=0)

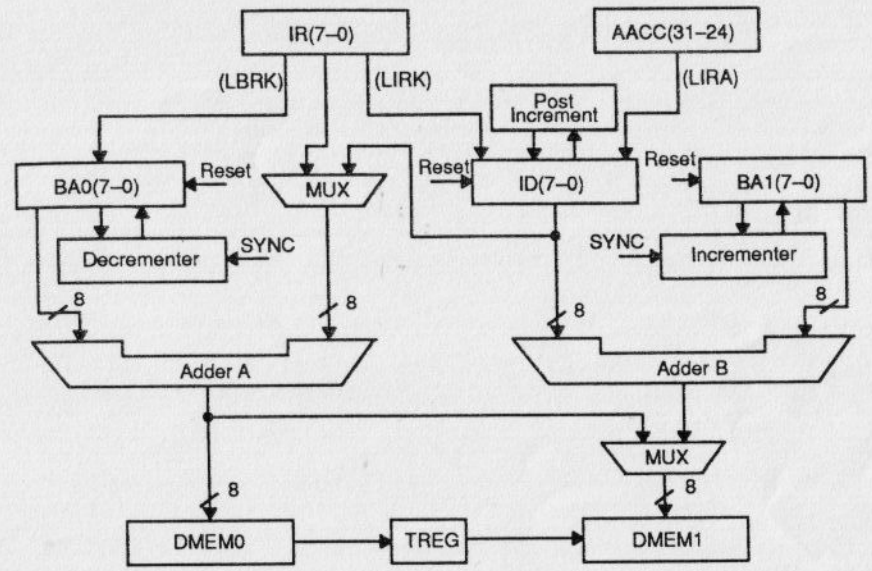
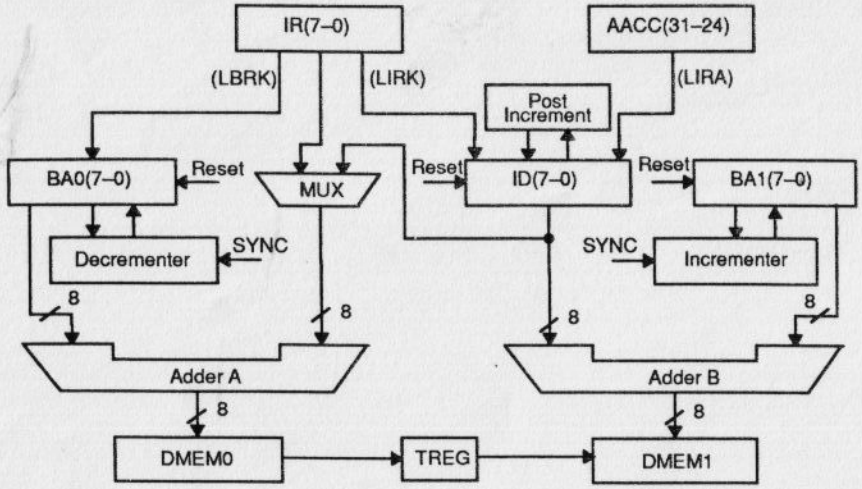


Figure 3–8. Data Memory Addressing Unit of the TMC57001 and TMS57002 (CAS=1)



To calculate an indirect address, the offset register (ID register) is added to a base address register. This result is the access address into data memory. Whether the BA0 register or BA1 register is used to calculate the indirect address is determined by the device and the data bank selected. The method of calculation of direct and indirect addresses for DMEM0 and DMEM1 access is summarized by 5700x device in Table 3–2.

For all 5700x devices, the DMEM0 indirect address is calculated by adding BA0 and ID in Adder A. The calculation of the DMEM1 indirect address is dependent upon the device. For the TMC57000 (and TMS57002 when CAS=0), the DMEM1 indirect address is calculated by adding BA0 and ID in Adder A. For TMC57001 (and TMS57002 when CAS=1), the DMEM1 indirect address is calculated by adding BA1 and ID in Adder B.

If an instruction uses indirect post–increment addressing, the ID register will be incremented by one after instruction execution. The ID register is used only for indirect addressing. The contents of the ID register can be set with the LIRK or LIRA instructions. The LIRK instruction loads the ID register with an immediate value. The LIRA instruction loads the ID register with the AACC contents. The ID register is reset to zero at the triggering edge of the SYNC signal.

To calculate a direct address, the BA0 register is added to bits 7 through 0 of the IR register (offset value from the base address register) in Adder A. If

DBP=0, DMEM0 is accessed with the calculated address; if DBP=1, DMEM1 is accessed. Direct memory access to DMEM1 is not available with the TMC57001 (and TMS57002 when CAS=1).

Table 3–2. Indirect/Direct Addressing Summary by Device

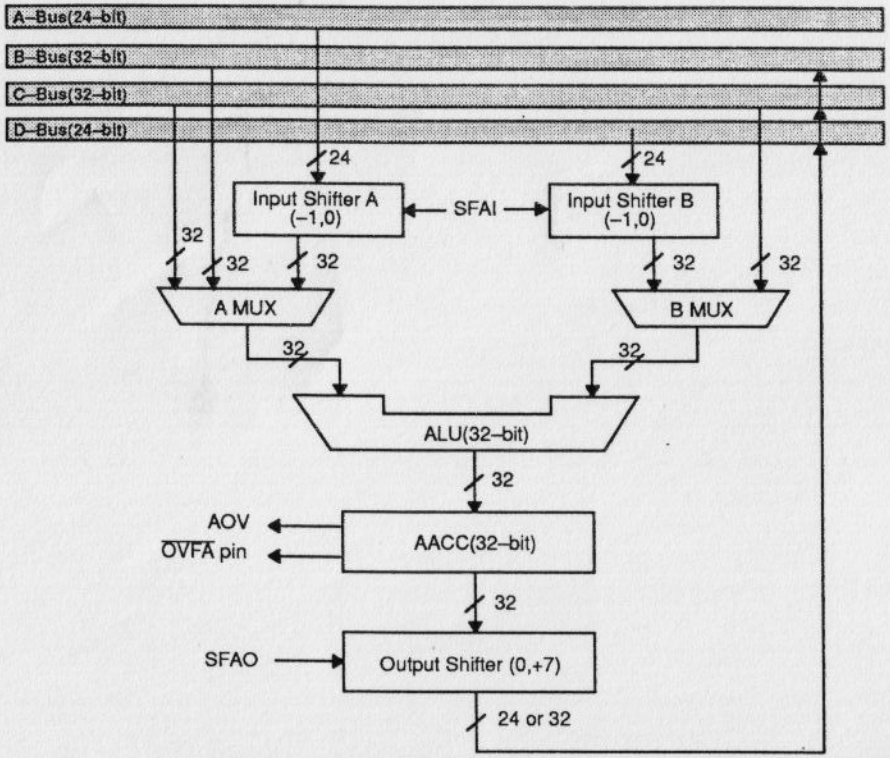
	DMEM0 Addressing(DBP=0)		DMEM1 Addressing(DBP=1)	
	Direct	Indirect	Direct	Indirect
TMC57000 and TMS57002 (CAS=0)	IR + BA0	ID + BA0	IR + BA0	ID + BA0
TMC57001 and TMS57002 (CAS=1)	IR + BA0	ID + BA0	NA	ID + BA1

Note: There is one exception to the above memory address calculations. For all 5700x devices, in the case of the ADDS, AMPY, AMAC, LD0T and STD1 instructions, the DBP setting is overridden. With the ADDS, AMPY and AMAC instructions, both DMEM0 and DMEM1 are accessed with indirect addressing. The DMEM0 indirect address is calculated by adding the BA0 register and the ID register, and the DMEM1 indirect address is calculated by adding the BA1 register and the ID register. With the LD0T instruction, DMEM0 is accessed with an indirect address which is calculated by adding the BA0 register and the ID register. With the STD1 instruction, DMEM1 is accessed with an indirect address which is calculated by adding the BA1 register and the ID register.

3.4 Arithmetic Logic Unit (ALU)

The Arithmetic Logic Unit (ALU) is one of the 5700x's two data processing units. The ALU contains a 32–bit arithmetic logic unit (ALU), a 32–bit accumulator (AACC), two input shifters and an output shifter. The ALU architecture is illustrated in Figure 3–9. The ALU implements a wide range of arithmetic and logical functions, the majority executing in a single clock cycle. The Multiplier Accumulator, the second of the 5700x's data processing units, is described in Section 3.5.

Figure 3–9. Arithmetic Logic Unit (ALU)



The ALU is a 32–bit general purpose arithmetic logic unit. Valid inputs are AACC, MACC and data values from DMEM0, DMEM1 and CMEM. The ALU can perform various arithmetic and logical operations. The result of the ALU is stored in the AACC.

All arithmetic calculations are performed on operands with fixed point notation. Refer to Figure 3–3 in Section 3.2.2 for a summary of the fractional data format.

Valid inputs to the ALU are 24 or 32–bit data values. Calculations with mixed bit–width inputs are allowed. Each of the two inputs, A and B, have a multiplexer to select 24 or 32–bit data. The 24–bit data is expanded to a 32–bit data value as specified by the control bit SFAI.

The 24–bit data inputs pass through Input Shifter A or Input Shifter B. The two input shifters are controlled by SFAI. If SFAI=0, there is no shifting and the 24–bit value is extended by eight lower zero–filled bits. If SFAI=1, the 24–bit value is shifted 1 bit right, the MSB is sign–extended by 1 bit and this 25–bit value is extended by seven lower zero–filled bits. SFAI can be set with the SFAI instruction.

These input shifters were especially implemented to prevent overflow during pre–addition of data samples in linear phase FIR filter applications when using the ADDS, AMAC and AMPY instructions. During execution of the ADDS, AMAC and AMPY instructions the setting of SFAI is ignored. By default the 24–bit operands are shifted 1 bit right in the same way as when SFAI=1.

The ADDS, AMAC, AMPY and MACS instructions are used exclusively for linear phase FIR filter calculations. The AMAC and AMPY instructions use both the ALU and the Multiplier Accumulator (MAC). The ALU performs pre–addition of two data samples. The MAC multiplies the result of the ALU with a coefficient memory value. At the ALU, overflow during pre–addition is prevented by dividing both 24–bit inputs by two with a 1–bit right shift at the input shifters. To restore the original value, at the input of the MAC, the 25–bit AACC value is multiplied by two with a 1–bit left shift. This is discussed in greater detail in Section 3.5 (Multiplier Accumulator). Refer to Section 5.4 for linear phase FIR filter applications.

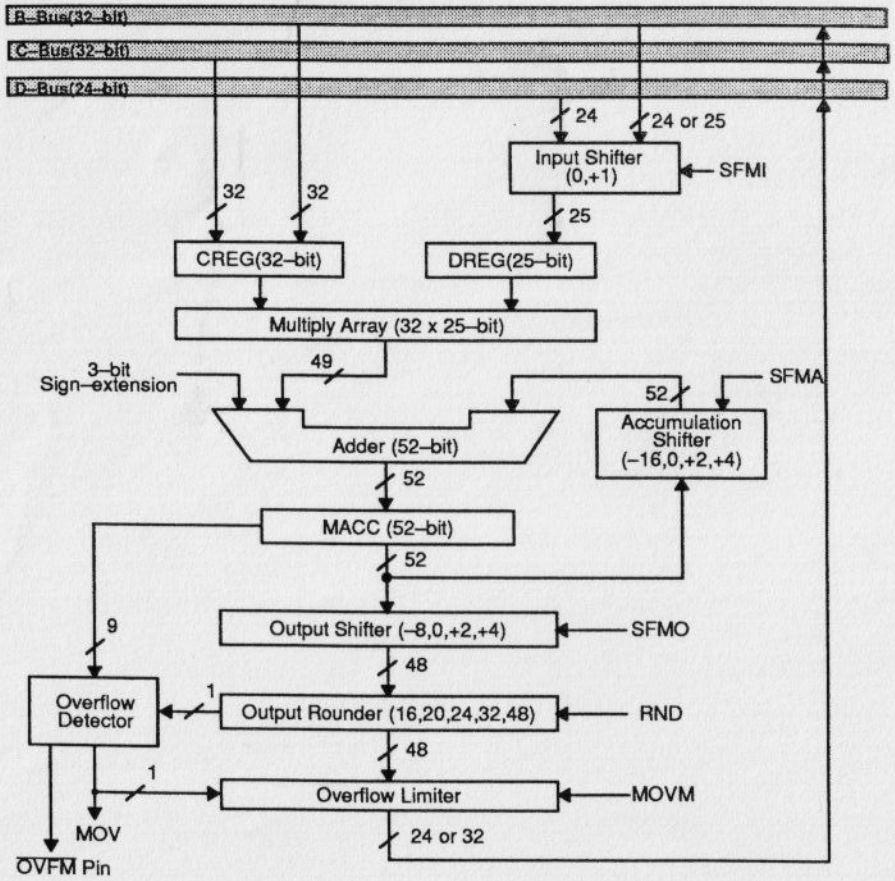
If an overflow condition is detected in the AACC, AOV is set. The value of the AACC remains in the overflow state. The branching instruction, BV, provides branching based upon the status of AOV. The status of AOV is output through the $\overline{\text{OVFA}}$ pin. AOV remains set until cleared by reset, synchronization, or the RAOV, INCD or BV instructions.

Before being transferred to a bus, the AACC value passes through the Output Shifter. If SFAO=0, there is no shifting. If SFAO=1, there is a 7–bit left logical shift. The contents of the AACC remain unchanged. SFAO can be set with the SFAO instruction. After passing through the Output Shifter either a 24 or 32–bit data value will be outputted. The output data bit–width is determined by the bit–width of the bus. In the case of 24–bit data output, only the higher 24 bits are outputted.

3.5 Multiplier Accumulator (MAC)

The Multiplier Accumulator (MAC) is the 5700x's main data processing unit. The MAC contains a multiplier, adder, accumulator(MACC), shifters, a rounder and an overflow limiter. The MAC architecture is illustrated in Figure 3–10. Both signed and unsigned multiplication are supported. The 5700x also has an arithmetic processing unit (ALU). Please refer to Section 3.4 for details on the ALU.

Figure 3–10. Multiplier Accumulator (MAC)



The 25 x 32–bit multiplier performs twos–complement multiplication. All multiplication is performed on operands in the fractional data format (signed data) or the unsigned data format (unsigned data). Refer to Section 3.2.2 for details on these data formats.

Signed multiplication of the following formats are supported: 24 x 32–bit and 25 x 32–bit. See Figure 3–11 and Figure 3–12. 24 x 32–bit multiplication is suitable for long tap FIR filter calculations of audio frequency signals. The 32–bit input is used for coefficient data and the 25–bit input is used for audio signal data. Signed multiplication with a 25–bit input value is used exclusively for implementing linear phase FIR filters. Refer to the discussion which follows for more details.

Unsigned multiplication (24–bit unsigned x 32–bit signed) is also supported. See Figure 3–13. Unsigned multiplication is available for implementing long word (40–bit DMEM by 32–bit CMEM) second order IIR filter applications. Refer to Section 5.7 for details. The MPYU (multiply unsigned) and MACU (multiply unsigned, accumulate) instructions perform unsigned multiplication.

Figure 3–11. (Signed 24–bit) x (Signed 32–bit) Multiplication

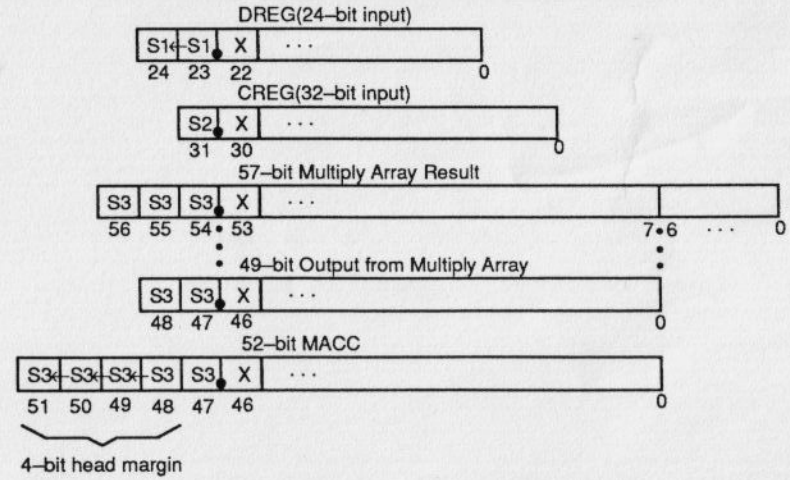


Figure 3–12. (Signed 25–bit) x (Signed 32–bit) Multiplication

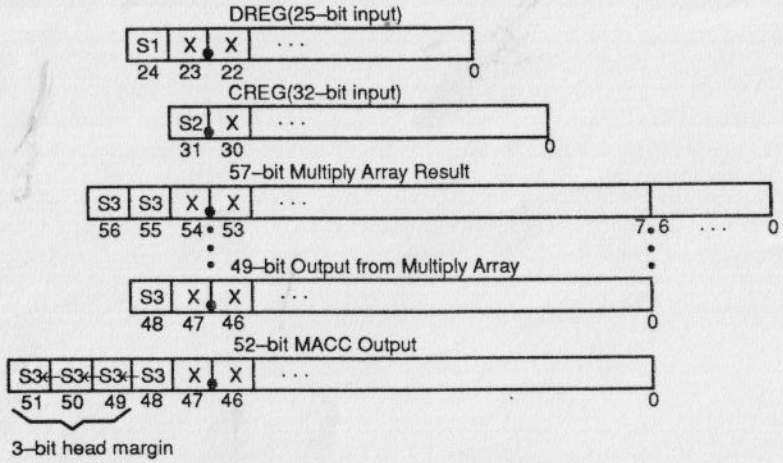
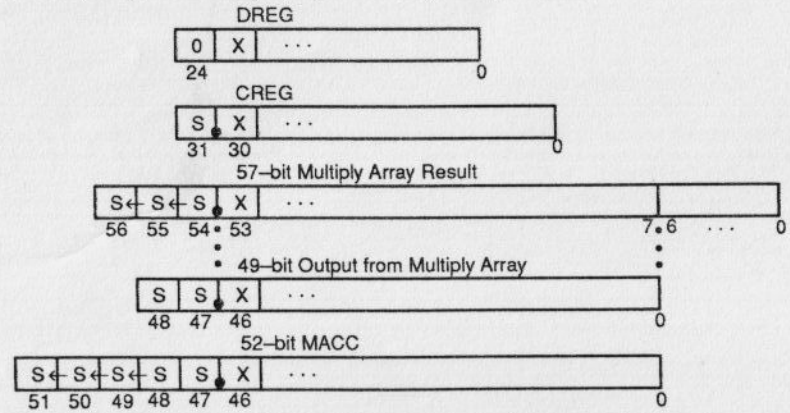


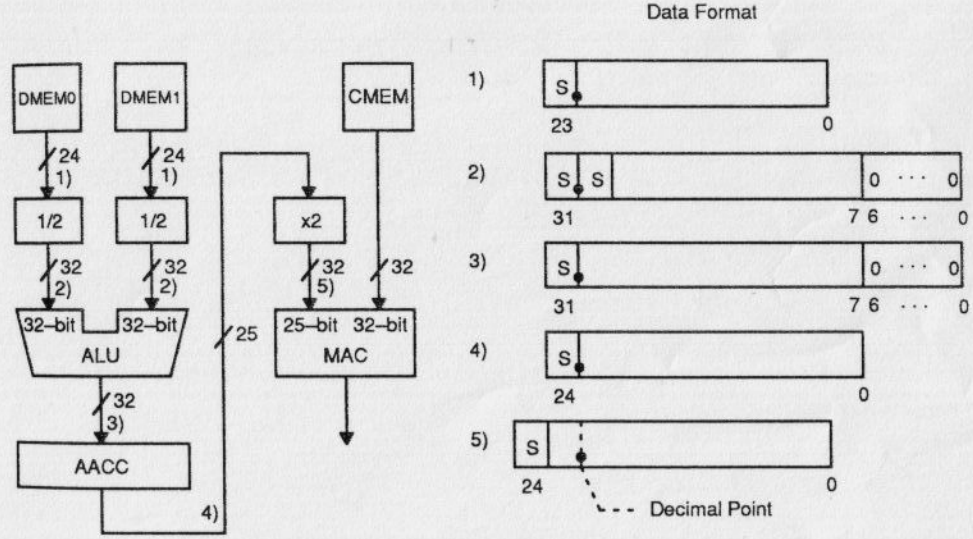
Figure 3–13. (Unsigned 24–bit) x (Signed 32–bit) Multiplication



The 5700x series offers unique architecture for the implementation of linear phase filters. Because the calculation unit of the 5700x has been designed as two separate units (the ALU and MAC), addition and multiplication calculations can be performed in parallel. Two instructions are available, AMAC and AMPY, which perform ALU and MAC calculations in parallel in one instruction cycle. These instructions, in addition to ADDS and MACS, make it possible to implement linear phase filters twice as fast as conventional DSP.

The flow of data through the 5700x device for linear phase FIR filter calculations when using the AMAC and AMPY instructions is illustrated in Figure 3–14. At time n, the ALU performs pre–addition of two data samples. The MAC multiplies the result of the ALU with a coefficient memory value. At the ALU, in order to avoid overflow when performing pre–addition of two data samples, the ALU Input Shifters divide both DMEM inputs by two. The ALU output (pre–addition result from time n–1), is used as an input to the MAC. At the MAC Input Shifter, the input is multiplied by two, thus restoring the ALU output value to its actual value.

Figure 3–14. Linear Phase FIR Filter Calculation



The ALU Input Shifters and MAC Input Shifter have been implemented especially for linear phase FIR filter calculations. The ALU Input Shifter is controlled by SFAI. If SFAI=0, there is no shift performed. If SFAI=1, the input is shifted 1 bit right, the MSB is sign–extended by 1 bit and this 25–bit value is extended by seven lower zero–filled bits. SFAI can be set with the SFAI instruction. During execution of the ADDS, AMAC and AMPY instructions, the setting of SFAI is ignored. By default SFAI=1 during the execution of these three instructions.

The MAC Input Shifter has been implemented exclusively for linear phase FIR filter calculations when using the AMAC, AMPY and MACS instructions. The MAC Input Shifter is not user controllable. The MAC Input Shifter is controlled

by SFMI. The input to the MAC Input Shifter is either a 24 or 25–bit value. Only with the AMAC, AMPY and MACS instructions is the input to the MAC Input Shifter a 25–bit value. When AMAC, AMPY and MACS are executed, SFMI=1 (1–bit left shift). For all other instructions, the input to the MAC Input Shifter is a 24–bit value and SFMI=0 (no shift).

The inputs to the MAC are loaded into DREG(25–bit) via the Input Shifter and CREG(32–bit). Note that the decimal point of DREG is after the second MSB, whereas the decimal point of the input and CREG is after the MSB.

In the case of 25–bit input when using the AMAC, AMPY and MACS instructions (SFAI=1), bits 31–7 are loaded from the B–Bus into DREG, passing through the Input Shifter without any processing. This has effectively multiplied the 25–bit input by two because the decimal point of the input is after the MSB whereas the decimal point of DREG is after the second MSB. See Figure 3–12.

In the case of all 24–bit input, SFAI=0(no shift). The decimal point of the input is after the MSB, whereas the decimal point of the 25–bit DREG is after the second MSB. Therefore, in the case of signed 24–bit input, the MSB is sign–extended by one bit in order to effectively shift the input zero bits. Similarly, for unsigned 24–bit input, the MSB is extended by one zero–filled bit. This 25–bit value is then loaded to DREG. Note that for 24–bit input coming from the B–Bus, bits 31–8 are loaded to the MAC Input Shifter.

With all 24 x 32–bit multiplication, the result of CREG(32–bit) multiplied by DREG(25–bit) is a 57–bit result with three sign bits. In the case of a signed 24–bit input (see Figure 3–11), the three sign bits, represented by S3, are the value of the sign bit as a result of the multiplication. In the case of an unsigned 24–bit input (see Figure 3–13), the sign bit of the value in CREG, S, has been sign–extended by two.

With both signed and unsigned 24–bit input, the MSB and the 7 LSBs are cut from the 57–bit Multiply Array result, and the result is a 49–bit output value from the Multiply Array. The input to the Adder is a 52–bit value. The 49–bit Multiply Array output value is sign–extended by 3–bits to expand the value to a 52–bit value. Thus the input to the MACC is a 52–bit value with a sign bit and a 4–bit head margin.

In Figure 3–12, the contents of MAC registers during 25 x 32–bit multiplication are illustrated. As discussed earlier, 25–bit input is possible only with the AMAC, AMPY and MACS instructions. Note that the 52–bit MACC has a 3–bit head margin. Because with 25–bit input, the data range is from –2 to 2 rather than from –1 to 1 as in the case of 24–bit input, the head margin is only 3 bits.

The inputs to the Adder are the Multiply Array result and the value of the MACC after having passed through the Accumulation Shifter. The Accumulation

Shifter shifts a value by –16, 0, 2 or 4 as defined by SFMA. SFMA can be set with the SFMA instruction. During accumulation, bits which exceed the head margin are lost and are not checked for overflow.

In the case of four instructions, DOML, SLMH, SLML and SMLD, the contents at the MACC are outputted to the D–Bus. With all other instructions the MACC value passes through the Output Shifter and the Output Rounder before being outputted. The Output Shifter shifts the MACC value by –8, 0, +2 or +4 as defined by SFMO. SFMO can be set with the SFMO instruction. The Output Rounder will round an output value to 16, 20, 24, 32 or 48–bits as defined by RND. RND can be set with the RND instruction.

An overflow condition is checked after the MACC value passes through the Output Shifter and again after the MACC value passes through the Output Rounder. MOV is set to one if overflow is detected. The MOV status is output at the $\overline{\text{OVFM}}$ pin. MOV can be reset to zero at reset, synchronization or with the RMOV or INCD instructions.

An overflow condition is checked for in the Overflow Detector. The 9 MSBs of the MACC are transferred to the Overflow Detector. The number of bits checked for overflow depends upon the mode of the Output Shifter. In the case of no shifting (SFMO=0), the 4–bit head margin and sign bit are checked for overflow. In the case of a 2–bit left shift (SFMO=1), in addition to the 4–bit head margin and sign bit, two additional bits are checked for overflow, and in the case of a 4–bit left shift (SFMO=2) four additional bits are checked for overflow. In the case of an 8–bit right shift (SFMO=3), no bits are checked for overflow.

The MOVM control bit controls the overflow limiter mode. If MOVM=1, the overflowed value is clipped to the most negative (80...00h) or most positive (7F...FFh) value. If MOVM=0, there is no clipping. MOVM is set with the SMOM instruction and reset to zero with the RMOM instruction.

After passing through the Output Shifter and Output Rounder, the output value goes to the Overflow Limiter. If MOVM=1, and overflow was detected in the Overflow Detector (MOV=1), the overflow value is clipped and either 7FFFFFFFFFFFh or 800000000000h is outputted from the Overflow Limiter. If overflow was not detected, the value from the Output Rounder is outputted. If MOVM=0, regardless of whether overflow was detected or not, the value from the Output Rounder is outputted. There are seven MACC output instructions: DOMH, DOML, SLMH, SLML, SMHD, SMLD and SMHC. Overflow detection is performed for all MACC output instructions, but only with the DOMH, SMHD and SMHC instructions does the MACC output pass through the Output Shifter and Output Rounder. Refer to Section 4.5 for details on the MACC output instructions.

3.6 System Control

System control of the 5700x processors is provided by the program counter, status registers, the external reset signal, and a repeat counter. The following sections describe these system control components in detail. In addition, pipeline operation and synchronization are described.

- Program Address Generation and Control
- Pipeline
- Status Registers
- Repeat Counter
- Reset
- Synchronization

3.6.1 Program Address Generation and Control

The 5700x has an 8–bit program counter (PC). The PC addresses internal program memory(PMEM) in fetching instructions. At the end of the instruction cycle, an instruction is loaded into the instruction register (IR). When the IR is loaded, the PC is ready to start the next instruction fetch cycle.

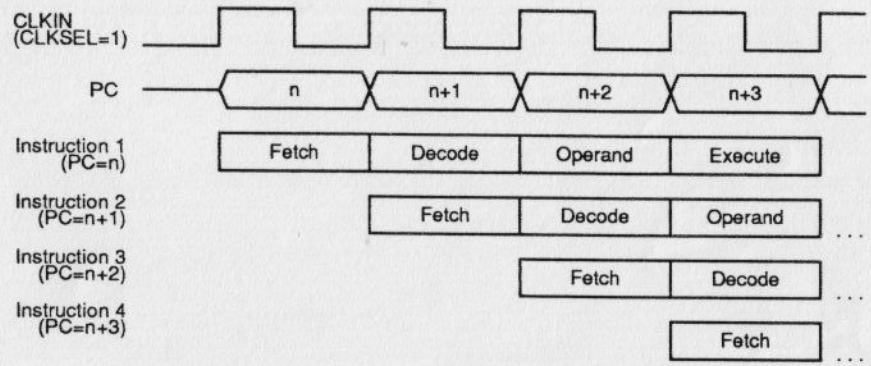
To start a new fetch cycle, the PC is loaded with PC+1 or with a branch address. When code is sequentially executed, the PC is loaded with PC+1. When a branch is executed, the PC is loaded with the immediate value with the branch instruction. In the case of conditional branches where the branch is not taken, the PC is incremented by one.

The PC is cleared to zero at hardware reset and at synchronization.

3.6.2 Pipeline

The instruction execution of the 5700x is pipelined to allow most instructions to execute at a rate of one instruction per clock cycle. The 5700x pipeline contains four sequential operations: fetch, decode, operand and execute. Thus during any given cycle, up to four different instructions can be active, each at a different stage in the pipeline. Figure 3–15 shows the operation of the four–level pipeline.

Figure 3–15. Four–level Pipeline Operation



The instructions are divided into 9 major classes based upon slight differences in their pipeline operations. Table 3–3 summarizes the instructions according to their categorization by class. This is followed by a further explanation of each class.

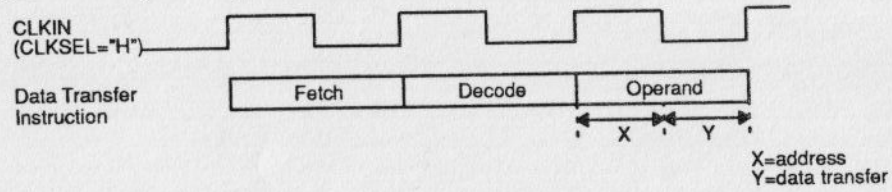
Table 3–3. Instruction Summary by Pipeline Operation

Class 1:	**Data Transfer Instructions**
	DIMH, DIML, DIS, DOMH, DOML, DOS, INCD, LBRK, LCAA, LCAK, LDPK, LDOT, LIRA, LIRK, LMHC, LMHD, LMLD, NOP1, NOP2, RAOV, RMOM, RMOV, RND, SACC, SACD, SCRM, SFAI, SFAO, SFMA, SFMO, SLMH, SLML, SMHC, SMHD, SMLD, SMOM, STD1, ZMAC TMS57002 only: SRBD, LPC, LPD
Class 2:	**ALU Instructions**
	ABS, ADD, ADDS, AND, CMPL, LACC, LACD, NEG, OR, SUB, XOR, ZACC
Class 3:	**Multiplication Instructions**
	MAC, MACS, MACU, MPY, MPYU, SFML, SFMR
Class 4:	**Parallel Instructions**
	AMAC, AMPY
Class 5:	**Branch Instructions**
	B, BGZ, BIOZ, BLZ, BNZ, BV
Class 6:	**Repeat Instruction**
	RPTK
Class 7:	**Conditional Load Instruction**
	LCAC
Class 8:	**Idle Instruction**
	IDLE
Class 9:	**External Memory Access Instructions (TMS57002 only)**
	RDE, REF, WRE

3.6.2.1 Class 1 (Data Transfer Instructions)

The pipeline operation of data transfer instructions does not include an execute operation as shown in Figure 3–16. The operand operation is divided into two parts. The first half of the cycle is reserved for the interpretation of the memory address. The second half of the cycle is used for the data transfer. In the case of immediate load instructions, since there is no need for interpreting a memory address, no operation is performed during the first half of the operand cycle.

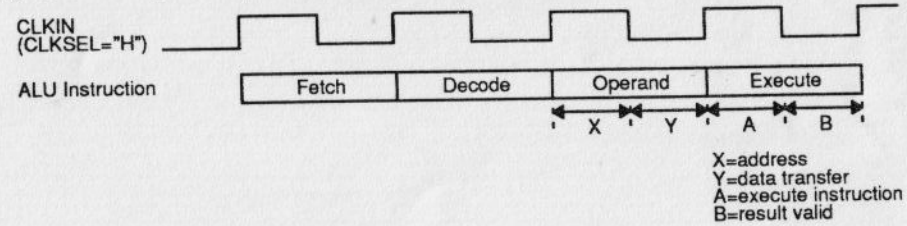
Figure 3–16. Pipeline Operation of Data Transfer Instructions



3.6.2.2 Class 2 (ALU Instructions)

The pipeline operation of ALU instructions includes an execute operation as illustrated in Figure 3–17 below. The instruction is executed in the first half of the cycle and the result is valid from the second half of the cycle. The operand operation is identical to that of data transfer instructions as explained for Class 1 instructions.

Figure 3–17. Pipeline Operation of ALU Instructions

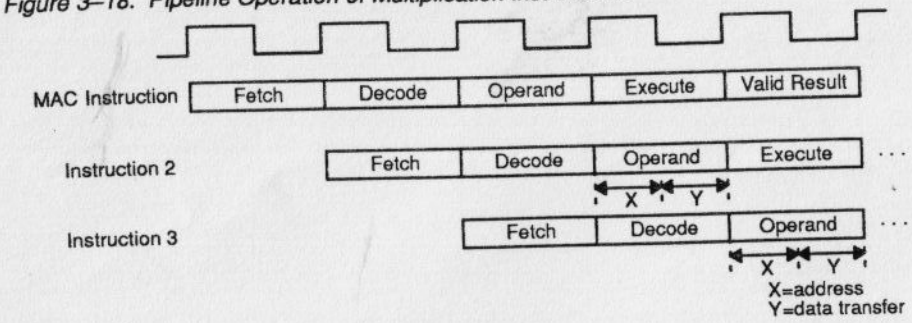


3.6.2.3 Class 3 (Multiplication Instructions)

The pipeline operation of multiplication instructions is illustrated in Figure 3–18. The operand operation is identical to that of ALU and data transfer instructions. Caution must be taken when using multiplication instructions because one complete cycle is required for execution. The result is not valid until the following cycle as shown in Figure 3–18. Use an instruction (ie. NOP1) to wait for the result to become valid as shown below:

```
MPY   D*+, C*+
MAC   D*+, C*+
MAC   D*+, C*+
NOP1
DOMH  2
```

Figure 3–18. Pipeline Operation of Multiplication Instructions



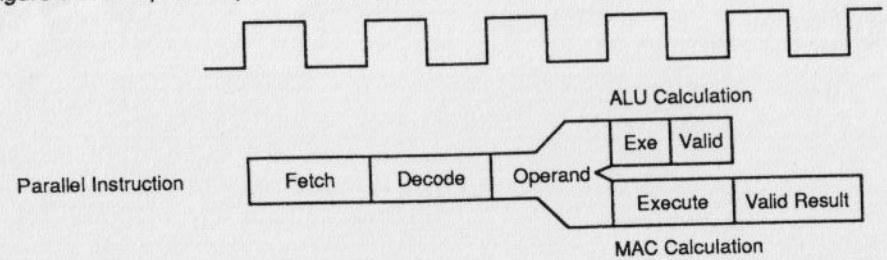
Note:

Caution must be taken in choosing the instruction that follows a class 3 (multiplication) or class 4 (parallel) instruction. A class 3/4 instruction can be followed only by an instruction which uses the class 3/4 instruction's result as an operand for accumulation (ie. MAC, MACS, AMAC or MACU). In all other cases, it takes two instruction cycles for the class 3/4 instruction's result to become valid. The result is valid in time for the second instruction following the class 3/4 instruction to use the result as an operand.

3.6.2.4 Class 4 (Parallel Instructions)

With the AMAC and AMPY instructions, ALU and MAC calculations are performed in parallel as shown in Figure 3–19 below.

Figure 3–19. Pipeline Operation of Parallel Instructions

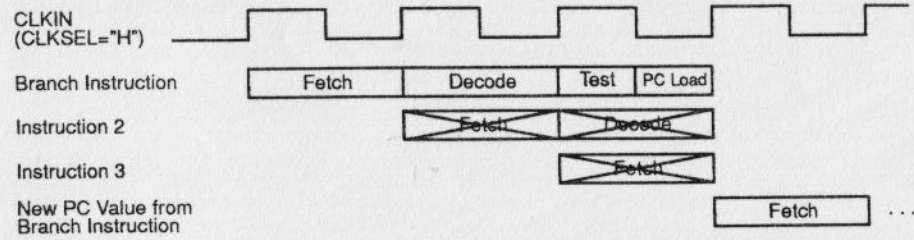


The ALU calculation is the same as class 2 instructions, and the MAC calculation is the same as class 3 instructions. Refer to Section 3.6.2.2 and Section 3.6.2.3 for details.

3.6.2.5 Class 5 (Branch Instructions)

When executing branch instructions if the branch is taken, the pipeline operation is interrupted as illustrated in Figure 3–20.

Figure 3–20. Pipeline Operation of Branch Instructions when Conditional Test is True



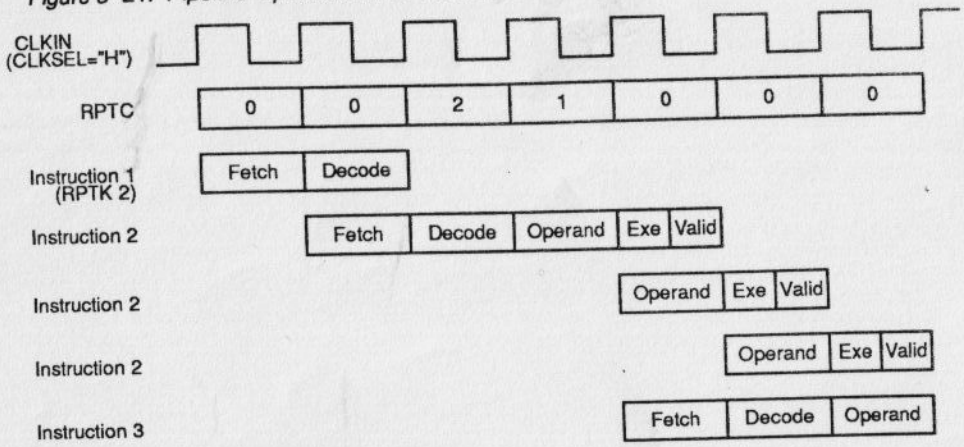
For all conditional branch instructions, during fetch and decode, pipeline operation continues as normal. During the test operation, if the conditional test is false, pipeline operation continues. If the conditional test is true, the branch PC address is loaded into the PC and the pre–fetch of the instructions that follow the branch instruction are aborted. In the next cycle, there is an instruction fetch operation at this new PC value.

With the branch unconditionally instruction, there is no conditional test as illustrated in Figure 3–20, but the rest of the pipeline operation is identical.

3.6.2.6 Class 6 (Repeat Instruction)

The pipeline operation of the RPTK instruction for a Class 2 instruction when RPTC=2 is illustrated in Figure 3–21. The instruction following the RPTK instruction is executed three times.

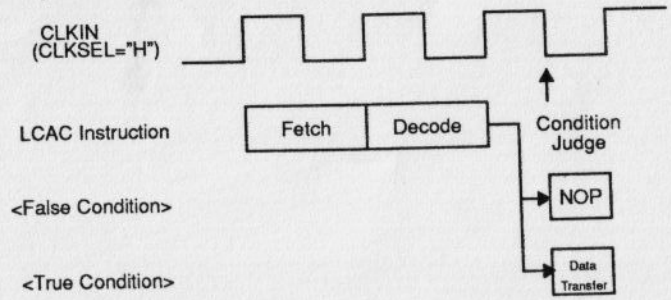
Figure 3–21. Pipeline Operation of RPTK Instruction for Class 2 Instruction



3.6.2.7 Class 7 (Conditional Instruction)

The pipeline operation of the LCAC instruction is illustrated in Figure 3–22.

Figure 3–22. Pipeline Operation of LCAC Instruction



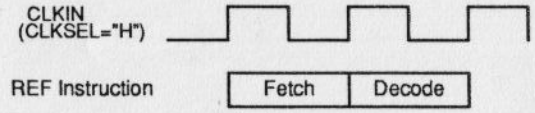
3.6.2.8 Class 8 (Idle Instruction)

After all pre–fetched instructions are completed, the processor is put into an idle state. Program execution is suspended until hardware reset or synchronization resets the PC to zero.

3.6.2.9 Class 9 External Memory Access Instructions (TMS57002 only)

The RDE and WRE instructions are identical to that of the data transfer instructions except that after the execute operation is complete, control passes to the external RAM interface for completion of processing of the instruction. REF is identical to the RDE and WRE instructions except there is no execute operation as illustrated in Figure 3–23.

Figure 3–23. Pipeline Operation of REF Instruction



3.6.3 Status Registers

The 5700x devices have two 24–bit status registers, ST0 and ST1. ST0 is primarily dedicated for serial I/O control, and ST1 is primarily dedicated for control of the ALU and Multiplier Accumulator. The assignment of control bits to ST0 and ST1 is illustrated in Figure 3–24. The contents of ST0 and ST1 should be downloaded before Program Memory (PMEM) contents are downloaded. Please refer to Section 3.7 for details on initializing ST0 and ST1.

The status register fields are described Table 3–4. They are arranged alphabetically by status register. Throughout this document, when reference is made to the value of a status register field for explanation purposes, the value will be represented in decimal notation.

Figure 3–24. Status Register Bit Assignment

Status Register 0

23–22	21	20–18	17–16	15	14	13	12	11	10–9	8	7	6	5	4	3	2	1	0
Reserved	SRAM	Reserved	M	SEL	WORD	CNS	PBCO	PLRO	SOM	FO	DIRO	PBCI	PLRI	Reserved	SIM	FI	DIRI	INCS

Status Register 1

23–22	21	20	19–18	17–15	14–13	12–11	10–9	8–7	6	5	4–3	2	1	0
Reserved	CAS	DBP	CRM	RND	Reserved	SFMO	Reserved	SFMA	MOV	MOVM	Reserved	SFAO	SFAI	AOV

Table 3–4. Status Register Field Definition

Field	Function
	Status Register 0 (ST0)
CNS	Suppress Right Channel Output Select. If CNS=0, the device operation is not affected. If CNS=1, the right channel serial output is suppressed at both the SO0 and SO1 ports. $\overline{RS}$ resets CNS to 0. Refer to Section 3.8.3.1 for more details.
DIRI	Receive Data LSB or MSB First Select. If DIRI=0, the two serial input ports (SI0 and SI1) will receive input audio data LSB first. If DIRI=1, the data will be received MSB first. $\overline{RS}$ resets DIRI to 0. Refer to Section 3.8.3.2 for more details.
DIRO	Transmit Data LSB or MSB First Select. If DIRO=0, the two serial output ports (SO0 and SO1) will transmit output audio data LSB first. If DIRO=1, the data will be transmitted MSB first. $\overline{RS}$ resets DIRO to 0. Refer to Section 3.8.3.3 for more details.
FI	Serial Input Format (Standard or I^2S) Select. FI sets the serial input timing to that of standard format or I^2S format. If FI=0, standard format is selected. The last data bit is received before the triggering edge of LRCKI. If FI=1, I^2S format is selected. The last data bit is received after the triggering edge of LRCKI. $\overline{RS}$ resets FI to 0. Refer to Section 3.8.3.4 for more details.
FO	Serial Output Format (Standard or I^2S) Select. FO sets the serial output timing to that of standard format or I^2S format. If FO=0, standard format is selected. The last data bit is transmitted before the triggering edge of LRCKO. If FI=1, I^2S format is selected. The last data bit is transmitted after the triggering edge of LRCKI. $\overline{RS}$ resets FO to 0. Please refer to Section 3.8.3.5 for more details.
INCS	Data Memory Base Address Register Control (TMS57002 only). If INCS=0, at the triggering edge of SYNC, the BA0 register will be decremented by one and the BA1 register will be incremented by one. With the TMS57002, if INCS=1, the BA0 register's decrementing function and the BA1 register's incrementing function are disabled. $\overline{RS}$ resets INCS to 0.
M	External RAM Addressing Space Select(TMS57002 only). If M=0, external memory has an addressing space of 64K. If M=1, the addressing space is 256K. If M=2, the addressing space is 1M.
PLRI	Left/right Channel Sub–frame Definition for LRCKI. If PLRI=0, the left channel data are received when LRCKI is low, and the right channel data are received when LRCKI is high. If PLRI=1, the left channel data are received when LRCKI is high, and the right channel data are received when LRCKI is low. $\overline{RS}$ resets PLRI to 0. Please refer to Section 3.8.3.8 for more details.
PLRO	Left/right Channel Sub–frame Definition for LRCKO. If PLRO=0, the left channel data are outputted when LRCKO is low, and the right channel data are outputted when LRCKO is high. If PLRO=1, the left channel data are outputted when LRCKO is high, and the right channel data are outputted when LRCKO is low. $\overline{RS}$ resets PLRO to 0. Please refer to Section 3.8.3.9 for more details.
PBCI	BCKI Active Edge Select. If PBCI=0, at the rising edge of BCKI, audio data from the serial input ports (SI0 and SI1) are shifted into the shift registers (SHIFTIN_0 and SHIFTIN_1). If PBCI =1, at the falling edge of BCKI, audio data are shifted into the shift registers. $\overline{RS}$ resets PBCI to 0. Please refer to Section 3.8.3.6 for more details.
PBCO	BCKO Active Edge Select. If PBCO=0, at the falling edge of BCKO, the output shift register contents (SHIFTOUT_0 and SHIFTOUT_1) are shifted to the serial output ports (SO0 and SO1). If PBCO =1, at the rising edge of BCKO, the data are shifted out. $\overline{RS}$ resets PBCO to 0. Please refer to Section 3.8.3.7 for more details.
SEL	External RAM Interface I/O Port Width(TMS57002 only). If SEL=0, the I/O port width is 4 bits and the ED7–4 pins are used for the I/O port. If SEL=1, the I/O port width is 8 bits.
SIM	Serial Input Data Format Select. If SIM=0, the data word length is 24 bits for both input ports (SI0 and SI1). If SIM=1, the data word length is 16 bits. $\overline{RS}$ resets SIM to 0. Please refer to Section 3.8.3.10 for more details.
SOM	Serial Output Data Format Select. The audio data format is determined by the data word length and the ratio of BCKO/LRCKO. The different formats available are summarized in Table 3–5. $\overline{RS}$ resets SOM to 0. Refer to Section 3.8.3.11 for more details.
SRAM	External RAM Select(TMS57002 only). If SRAM=0, DRAM is selected as external RAM. If SRAM=1, SRAM is selected as external RAM.

Table 3–4. Status Register Field Definition (Continued)

Field	Function
	Status Register 0 (ST0) (Continued)
WORD	External RAM Data Width Select(TMS57002 only). If WORD=0, the data width of data which is stored in external RAM is 16 bits. If WORD=1, the data width is 24 bits.
	Status Register 1 (ST1)
AOV	AACC (ALU Accumulator) Overflow Flag. AOV is set to one when an overflow occurs in the AACC. The inverted status of AOV is output at the $\overline{OVFA}$ pin. The BV instruction provides branching on the status of AOV. AOV is reset to zero at reset, synchronization or with the BV, INCD or RAOV instructions.
CAS	TMS57002 Addressing Mode Select(TMS57002 only). If CAS=0, the Data Memory addressing unit is identical to that of the TMC57000. If CAS=1, The Data Memory addressing unit is identical to that of the TMC57001.
CRM	Coefficient Memory Read Mode. This bit defines the data format for coefficient memory read operation. If CRM=0, the coefficient memory data is read as a 32–bit value. If CRM=1, the higher 16 bits are read and the lower 16 bits of the output are zero–filled. If CRM=2, the lower 16 bits of coefficient memory are read as higher 16 bits and the lower 16 bits of the output are zero–filled. Only the coefficient memory read operation is affected by this control bit. CRM is set by the SCRM instruction. $\overline{RS}$ resets CRM to zero.
DBP	Data Memory Bank Select. If DBP=0, DMEM0 is selected for data memory operations. If DBP=1, DMEM1 is selected. When using the ADDS, AMAC, AMPY, LD0T and STD1 instructions, the setting of DBP is overridden. DBP is set by the LDPK instruction and cleared to zero at reset or synchronization.
MOV	MACC (MAC Accumulator) Overflow Flag. MOV is set to one when an overflow occurs in the MACC. The inverted status of MOV is output at the $\overline{OVFM}$ pin. MOV is reset to zero at synchronization, reset or with the RMOV or INCD instructions.
MOVM	MACC Overflow Mode. When overflow occurs in the MACC, if MOVM=1,the overflowed data from the MACC is clipped to the largest positive(7F...Fh) or negative(80...0h) value. If MOVM=0, when overflow occurs there is no clipping. MOVM is set with the SMOM instruction. MOVM is reset to zero with the RMOM instruction.
RND	MAC Output Rounder Mode Select. MAC output passes through the Output Rounder before being outputted. The MACC value is rounded to a 16, 20, 24 or 32 or 48–bit value as defined by the RND control bit. Refer to the RND instruction description for details. RND is reset to zero at reset.
SFAI	ALU 24–bit Input Shifter Mode Select. If SFAI=0, the 24–bit input value is not shifted. The value is expanded to a 32–bit value by extending the value with 8 zero–filled bits. If SFAI=1, the 24–bit input value will be shifted 1 bit to the right with a 1 bit sign extension and extended by 7 zero–filled bits. With the AMAC, AMPY and ADDS instructions the setting of SFAI is ignored. SFAI is set with the SFAI instruction. Refer to the SFAI instruction description for more details. SFAI is reset to zero at reset.
SFAO	ALU Output Shifter Mode Select. If SFAO=0, the ALU output is not shifted. IF SFAO=1, the ALU output is shifted 7–bits to the left. SFAO is set with the SFAO instruction. Please refer to the SFAO instruction description for more details. SFAO is reset to zero at reset.
SFMA	MAC Accumulation Shifter Mode Select. SFMA sets the shifting mode of the Accumulation Shifter. The MACC value passes through the Accumulation Shifter before accumulation. The MACC value is shifted by 0, +2, +4 or –16 as defined by the SFMA control bit. Refer to the SFMA instruction description for more details. SFMA is reset to zero at reset.
SFMO	MAC Output Shifter Mode Select. SFMO set the shifting mode of the Output Shifter. The Output Shifter shifts the MACC value by –8, 0, +2 or +4 as defined by the SFMO control bit. Refer to the SFMO instruction description for more details. SFMO is cleared to zero at reset.

Table 3–5. Serial Output Data Format Summary

Data Format (SOM)	SOM=0	SOM=1	SOM=2	SOM=3
Word Bit Length	24	16	16	16
BCKO/LRCKO	64	64	48	32

3.6.4 Repeat Counter

RPTC is an 8–bit repeat counter, which, when loaded with a number N, causes the next single instruction to be executed N+1 times. The RPTC register is loaded with the immediate value specified by the RPTK instruction. RPTC is cleared by reset.

The repeat function is available for instructions that perform addition, subtraction, multiplication, shifting and loading from the MAC and ALU to internal memory. Refer to the instruction description of a particular instruction (Section 4.5) to determine whether the repeat function is available or not.

3.6.5 Reset

Reset ($\overline{RS}$) is used for hardware initialization. Because the processor is in a random state at powerup, reset should be applied to initialize the hardware.

Driving the $\overline{RS}$ signal low causes the 5700x to terminate execution, forces the program counter to zero and resets various registers and control bits. A reset signal must be asserted low for at least three clock signals to guarantee a reset of the device.

Upon setting the $\overline{RS}$ signal low for five clock cycles, the following actions take place:

- ❑ The following registers are reset to zero:
  - PC
  - CA
  - ID
  - BA0
  - BA1
  - ST0 control bits:

    INCS, DIRI, FI, SIM, PLRI, PBCI, DIRO, FO, SOM, PLRO, PBCO, CNS
  - ST1 control bits:

    AOV, SFAI, SFAO, MOVM, MOV, SFMA, SFMO, RND, CRM, DBP
- ❑ SO0 and SO1 put into a high impedance state.
- ❑ The 16 host interface EMPTY flags are set to zero, and as result the EMPTY pin is set to one.

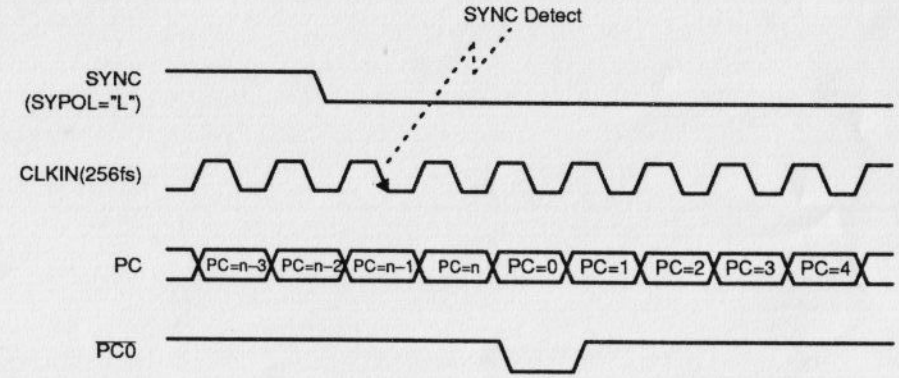
After reset operations are completed, $\overline{RS}$ should be set high. After the high state of $\overline{RS}$ is detected at the falling edge of CLKIN, the next 5700x process will begin after a one and a half clock delay. For information on operations that should follow reset, refer to Section 5.2, Processor Initialization.

3.6.6 Synchronization

The SYNC input signal is used for device synchronization. If SYPOL="L", synchronization is triggered on the falling edge of SYNC. If SYPOL="H", synchronization is triggered on the rising edge of SYNC.

After the triggering edge of SYNC has been detected by CLKIN, it takes one and a half clock cycles to ensure that the device has been synchronized. This is illustrated in Figure 3–25 where the PC is reset to zero one and a half clock cycles after the synchronization state has been detected. While PC=0, the $\overline{PC0}$ signal is set to active low. Note that at PC=n, the fetch operation of instruction n is executed. Refer to Section 3.6.2, Pipeline Operation, for more details.

Figure 3–25. Synchronization Timing



Below is a list of all actions which take place at synchronization:

- PC=0
- CA=0
- ID=0
- If INCS=0,
  - BA0 = BA0 – 1
  - BA1 = BA1 + 1
- AOV = 0 (ST1 control bit)
- MOV = 0 (ST1 control bit)

3.7 Host Interface

Via the host interface, data are transferred between the 5700x and the host controller. The types of data transfers available are listed below:

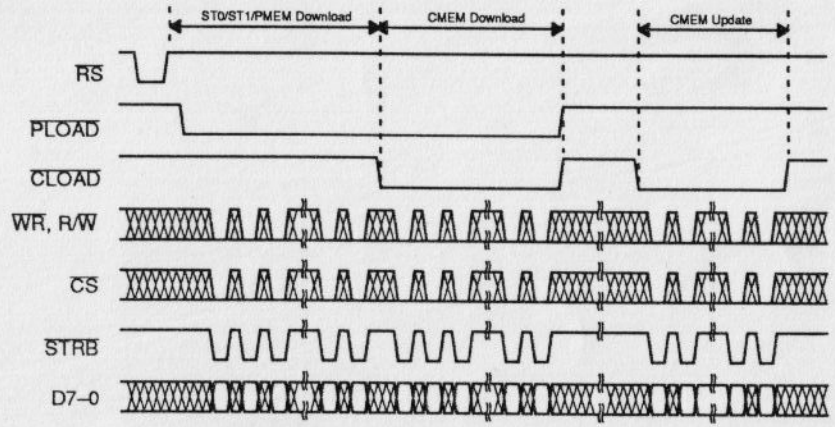
- Status register (ST0 and ST1) and program memory (PMEM) download
- Coefficient memory (CMEM) download
- CMEM update
- CMEM/DMEM output to host controller (TMS57002 only)

The download and update operations of the host interface are controlled by the settings of $\overline{\text{PLOAD}}$ and $\overline{\text{CLOAD}}$ when $\overline{\text{WR}}$(R/$\overline{\text{W}}$ for the TMS57002) is set low. The 5700x has four modes of operation which are summarized in Table 3–6. The host interface timing for ST0/ST1/PMEM download, CMEM download and CMEM update is illustrated in Figure 3–26.

Table 3–6. Host Interface Modes

$\overline{\text{PLOAD}}$	$\overline{\text{CLOAD}}$	MODE
1	1	Normal operation
0	1	ST0/ST1/PMEM download
0	0	CMEM download
1	0	CMEM update

Figure 3–26. Host Interface Timing

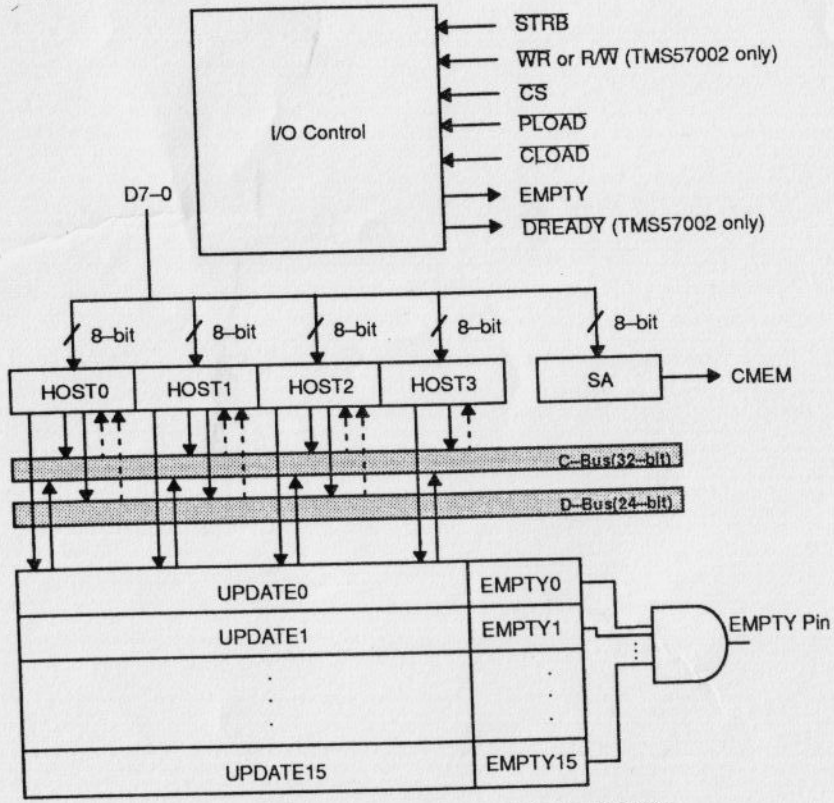


Because during normal operation, download and update operations are not possible, $\overline{\text{WR}}$ (R/$\overline{\text{W}}$ for the TMS57002) can be kept low at all times. There is

one exception to this. With the TMS57002, if R/$\overline{W}$ is set high during normal operation, CMEM and DMEM output to the host controller is possible. Please refer to Section 3.7.4 for more details.

The architecture of the host interface is illustrated in Figure 3–27. The pins, registers and instructions that are related to the host interface operation are summarized on page 3-39.

Figure 3–27. Host Interface



Note: Hyphenated arrow line available only for the TMS57002

Table 3–7. Host Interface Pins

Name	Description
$\overline{CLOAD}$	CMEM download/update mode control signal
$\overline{CS}$	$\overline{STRB}$ and $\overline{WR}$ (R/$\overline{W}$ for the TMS57002) enable signal
$\overline{DREADY}$	Handshaking signal for LPC/LPD instructions
EMPTY	Indicates all UPDATE registers are empty
$\overline{PLOAD}$	PMEM download mode control signal
$\overline{STRB}$	Data strobe signal
$\overline{WR}$	Write signal (all devices except TMS57002)
R/$\overline{W}$	Read/write signal (TMS57002 only)

Table 3–8. Host Interface Registers

Name	Description
EMPTYx ($0 \leq x \leq 15$)	UPDATEx empty flag
UPDATEx ($0 \leq x \leq 15$)	32–bit registers for CMEM update word
HOSTx ($0 \leq x \leq 3$)	8–bit registers for 24/32–bit word
SA	Start address for CMEM update

Table 3–9. Host Interface Instructions

Mnemonic	Description
LPC (TMS57002 only)	Load CMEM word to HOST registers
LPD (TMS57002 only)	Load DMEM word to HOST registers

3.7.1 Status Register/Program Memory Download

The host controller downloads the contents of the status registers (ST0 and ST1) and program memory (PMEM) via the host interface when $\overline{PLOAD}$ is low and $\overline{CLOAD}$ is high. Up to 258 words (ST0, ST1 and 256 words of PMEM) can be downloaded. To download one 24–bit ST0, ST1 or PMEM word, three bytes must be downloaded in succession through the D7 through D0 pins.

At the rising edge of $\overline{STRB}$ ($\overline{CS}$ and $\overline{WR}$ (R/$\overline{W}$ for the TMS57002) must be low while $\overline{STRB}$ is low), the data at the D7 through D0 pins are latched into a HOST register. The most significant byte is received first and latched into HOST0. The following two bytes are latched into HOST1 and HOST2, respectively. After the least significant byte is latched into HOST2, the contents of these registers are transferred to ST0, ST1 or PMEM via the D–Bus.

Note:

If 3 byte words are not downloaded, the bytes will not be latched into the HOST registers in proper sequence. For most applications CMEM download follows ST0/ST1/PMEM download. In this case, if an error occurs during ST0/ST1/PMEM download, the data will also be latched improperly during CMEM download.

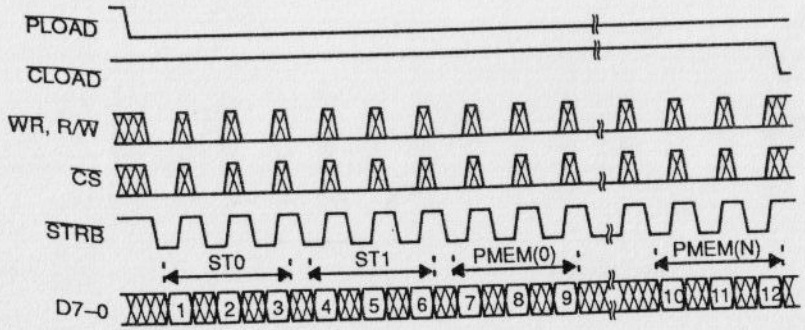
Refer to Figure 3–28 for ST0, ST1 and PMEM download timing. Below is a list of the order in which ST0, ST1 and PMEM bytes are downloaded.

1. ST0, bits 23–16 (higher byte)
2. ST0, bits 15–8
3. ST0, bits 7–0 (lower byte)
4. ST1, bits 23–16 (higher byte)
5. ST1, bits 15–8
6. ST1, bits 7–0 (lower byte)
7. PMEM(0), bits 23–16 (higher byte)
8. PMEM(0), bits 15–8
9. PMEM(0), bits 7–0 (lower byte)

.
.
.

10. PMEM(N), (N≤255), bits 23–16 (higher byte)
11. PMEM(N), (N≤255), bits 15–8
12. PMEM(N), (N≤255), bits 7–0 (lower byte)

Figure 3–28. ST0/ST1/PMEM Download Timing



3.7.2 Coefficient Memory Download

The host controller downloads the coefficient memory (CMEM) contents via the host interface when $\overline{\text{PLOAD}}$ and $\overline{\text{CLOAD}}$ are set low. Up to 256 words can be downloaded. To download one 32–bit CMEM word, four bytes must be downloaded in succession through the D7 through D0 pins.

At the rising edge of $\overline{\text{STRB}}$ ($\overline{\text{CS}}$ and $\overline{\text{WR}}$ (R/$\overline{\text{W}}$ for the TMS57002) must be low while $\overline{\text{STRB}}$ is low), the data at the D7 through D0 pins are latched into a HOST register. The most significant byte is received first and latched into HOST0. The following three bytes are latched into HOST1, HOST2 and HOST3, respectively. After the last byte is latched into HOST3, the contents of these registers are transferred to CMEM via the C–Bus and the CA register is incremented by one. Only if exactly 256 words are downloaded will the value of the CA register be reset to zero upon completion of CMEM download.

Note:

If 4 byte words are not downloaded, the bytes will not be latched into the HOST registers in proper sequence.

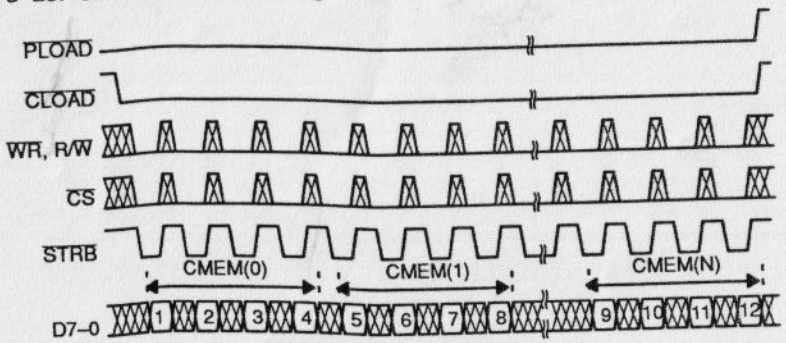
Refer to Figure 3–29 for CMEM download timing. Below is a list of the order in which CMEM bytes are downloaded.

1. CMEM(0), bits 31–24 (higher byte)
2. CMEM(0), bits 23–16
3. CMEM(0), bits 15–8
4. CMEM(0), bits 7–0 (lower byte)
5. CMEM(1), bits 31–24 (higher byte)
6. CMEM(1), bits 23–16
7. CMEM(1), bits 15–8
8. CMEM(1), bits 7–0 (lower byte)

.
.
.

9. CMEM(N), N≤255, bits 31–24 (higher byte)
10. CMEM(N), N≤255, bits 23–16
11. CMEM(N), N≤255, bits 15–8
12. CMEM(N), N≤255, bits 7–0 (lower byte)

Figure 3–29. CMEM Download Timing



3.7.3 Coefficient Memory Update

The 5700x enters coefficient memory (CMEM) update mode when $\overline{\text{PLOAD}}$ is set to high and $\overline{\text{CLOAD}}$ is set to low. The host interface can update up to sixteen 32–bit coefficient memory words. After $\overline{\text{CLOAD}}$ is set low, an 8–bit value at the D7 through D0 pins is latched into the SA register on the first rising edge of $\overline{\text{STRB}}$. This value is the CMEM address at which CMEM update will begin. After the specified starting address value is latched into the SA register, a sequence of up to sixteen 32–bit CMEM words are latched into the UPDATE registers via the HOST registers.

The method of latching the CMEM words into the HOST registers is identical to that for coefficient memory download. At the rising edge of $\overline{\text{STRB}}$ ($\overline{\text{CS}}$ and $\overline{\text{WR}}$ (R/$\overline{\text{W}}$ for the TMS57002) must be low while $\overline{\text{STRB}}$ is low), the data at the D7 through D0 pins are latched into a HOST register. The most significant byte is received first and latched into HOST0. The following three bytes are latched into HOST1, HOST2 and HOST3, respectively.

After the least significant byte is latched into HOST3, the contents of the HOST registers are loaded into an UPDATE register. The first CMEM update word is loaded into UPDATE0. The following CMEM update words are loaded into the UPDATE registers in sequential order.

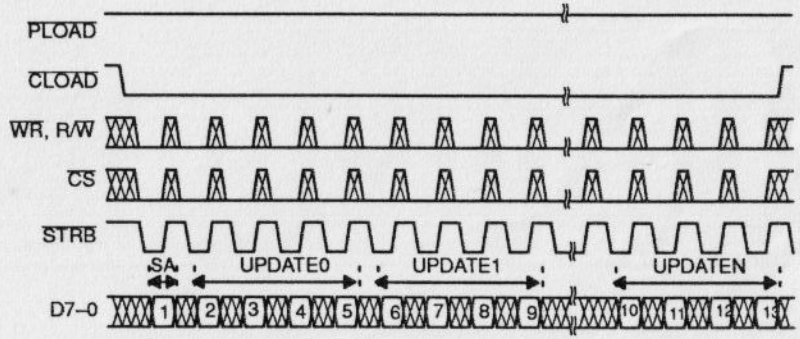
Refer to Figure 3–30 for CMEM update timing. Below is a list of the order in which CMEM bytes are stored in the UPDATE registers.

1. CMEM update starting address (1 byte)
2. UPDATE0, bits 31–24 (higher byte)
3. UPDATE0, bits 23–16
4. UPDATE0, bits 15–8
5. UPDATE0, bits 7–0 (lower byte)
6. UPDATE1, bits 31–24 (higher byte)
7. UPDATE1, bits 23–16
8. UPDATE1, bits 15–8
9. UPDATE1, bits 7–0 (lower byte)

.
.
.

10. UPDATEN, N≤15, bits 31–24 (higher byte)
11. UPDATEN, N≤15, bits 23–16
12. UPDATEN, N≤15, bits 15–8
13. UPDATEN, N≤15, bits 7–0 (lower byte)

Figure 3–30. CMEM Update Timing



When loading of the UPDATE registers is complete, $\overline{\text{CLOAD}}$ is set high. For every CMEM read access after $\overline{\text{CLOAD}}$ has been set high, the access address into CMEM is compared to the contents of the SA register. When the addresses are the same, coefficient update begins. The contents of UPDATE0 are written to CMEM at the address specified by the SA register. In order to avoid a bus collision, there is no CMEM read access but rather the contents of UPDATE0 are used as an operand for the current instruction.

Note:

If there is more than one instruction in program memory with an operand that has the same address value as the SA register value, it is uncertain as to where CMEM update will begin.

Coefficient update continues every time an instruction with CMEM read access is executed until all the UPDATE registers are empty. At every instruction

that is executed which has a read access to CMEM, the next UPDATE register value in sequential order is used as an operand. The UPDATE register value is then written to CMEM at the address specified by the executed instruction. After CMEM has been updated with a particular UPDATE register value, the corresponding EMPTY flag is set. When all the flags are set, thus indicating the completion of CMEM update, the EMPTY pin is set to high.

Note:

The value of CRM is ignored during CMEM update and by default, CRM=0. After CMEM update is complete, the value of CRM will be effective again.

The CMEM address location at which the UPDATE register value is written is different for direct addressing and indirect addressing. In the case of direct addressing, the UPDATE register value is written to CMEM at the specified direct address. In the case of indirect addressing, the UPDATE register value is written to CMEM at the value of the CA register.

Please take note of the following case in order to avoid errors during coefficient update. For example, in the following sequence of instructions,

```
MAC D*+, C*
MPY D*+, C*+
MAC D*+, C(9)
MAC D*+, C*
```

whereby an instruction with indirect addressing (no post increment) is followed by an instruction with indirect addressing (post increment), the UPDATE register values used to update CMEM when executing these two instructions will be written at the same CMEM address location, the value of the CA register. For the instruction with direct addressing, the update value is written to CMEM at the direct address specified. For example if,

```
CA=10
SA=10
UPDATE0=00000001
UPDATE1=00000002
UPDATE2=00000003
UPDATE3=00000004
```

then the contents of CMEM and the CA register after the execution of each line are as follows:

```
MAC D*+, C*
    CMEM(9):  FFFFFFFF
    CMEM(10): 00000001
    CMEM(11): FFFFFFFF
    CA     :  10
```



```
MPY D*+, C*+
    CMEM(9):  FFFFFFFF
    CMEM(10): 00000002
    CMEM(11): FFFFFFFF
    CA     :  11
MAC D*+, C(9)
    CMEM(9):  00000003
    CMEM(10): 00000002
    CMEM(11): FFFFFFFF
    CA     :  11
MAC D*+, C*
    CMEM(9):  00000003
    CMEM(10): 00000002
    CMEM(11): 00000004
    CA     :  11
```

3.7.4 CMEM/DMEM Output to Host Controller (TMS57002 only)

This section is only relevant to the TMS57002. The host interface of the TMS57002 has bidirectional capabilities. Data from CMEM and DMEM can be made available to the host controller. The LPC/LPD instructions read the contents of the addressed coefficient/data memory location and load one word of data to the HOST registers, the most significant byte being loaded to HOST0. The data in the four HOST registers becomes valid at pins D7 through D0 at the falling edge of STRB. The HOST registers are outputted in order from HOST0 through HOST3.

Note:

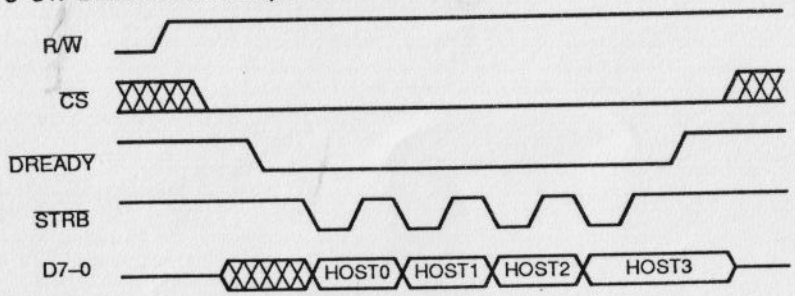
All four 8–bit HOST registers are outputted to pins D7 through D0. This is true not only for 32–bit CMEM words but also for 24–bit DMEM words. In the case of a 24–bit DMEM word, the contents of HOST3 are invalid.

The timing sequence required to read one CMEM/DMEM word is summarized below. Please refer to Figure 3–31 for the timing diagram.

- ❑ Set R/$\overline{W}$ high.
- ❑ Execute LPC/LPD.
- ❑ After LPC/LPD execution is complete, $\overline{DREADY}$ is set low to signal that one word of data is valid at the HOST registers.
- ❑ The contents of the four HOST registers become valid at pins D7 through D0 at the falling edge of $\overline{STRB}$ from most significant byte (HOST0) to least significant byte (HOST3).
- ❑ $\overline{DREADY}$ is set high after the rising edge of $\overline{STRB}$. At the falling edge of $\overline{STRB}$, the contents of HOST3 are transferred to pins D7 through D0.

Please make note that $\overline{CS}$ must be set low before $\overline{DREADY}$ is set low and must remain low until after $\overline{DREADY}$ is set high.

Figure 3–31. DMEM/CMEM Output to Host Controller



3.8 Serial Port

The 5700x series features a full–duplex on–chip serial port that provides direct communication with A/D converters, D/A converters and digital audio interface devices. The serial port may also be used for intercommunication between 5700x series processors in multiprocessing applications.

This serial I/O port has been designed especially for the transmission of audio data. In one sampling period, two channels of data (left and right) are transmitted. This is reflected in the serial I/O port architecture in which there is a pair of left and right channel registers associated with every port (refer to Figure 3–32 and Figure 3–33). A pair of left and right channel registers are double buffered to a shift register. This architecture is common to the two serial input ports (SI0, SI1) and the two serial output ports (SO0, SO1).

The transmit and receive sections of the serial port are implemented separately to allow independent transmit and receive operations. The LRCKI/LRCKO clock is the synchronization signal for one frame of input/output data. A frame is defined as one word of left and right channel data. The BCKI and BCKO clocks receive and transmit data, respectively.

The pins, registers and control bits that control serial port operation are summarized on the following pages. In addition, syntax specific to the serial I/O port is explained.

Figure 3–32. Serial Input Port

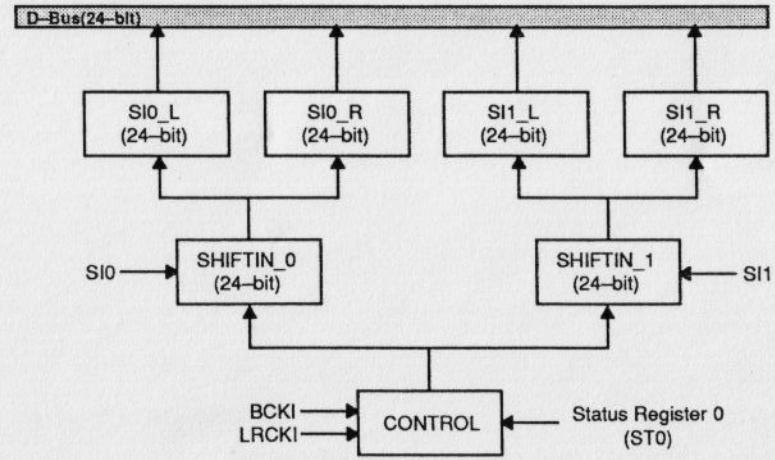


Figure 3–33. Serial Output Port

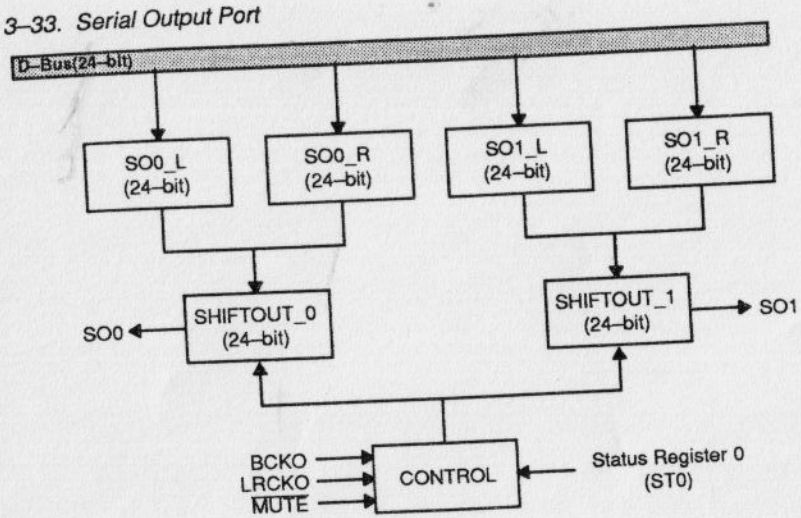


Table 3–10. Serial Port Pins

Name	Description
SI0/SI1	Received serial data signal at input port 0/1.
SO0/SO1	Transmitted serial data signal at output port 0/1.
BCKI	Receive data clock signal at SI0 and SI1.
BCKO	Transmit data clock signal at SO0 and SO1.
LRCKI	Left/right channel framing synchronization signal for SI0 and SI1.
LRCKO	Left/right channel framing synchronization signal for SO0 and SO1.

Table 3–11. Serial Port Registers

Name	Description
SHIFTIN_0	Input data shift register for SI0 port.
SHIFTIN_1	Input data shift register for SI1 port.
SHIFTOUT_0	Output data shift register for SO0 port.
SHIFTOUT_1	Output data shift register for SO1 port.
SI0_L/SI0_R	Left/right channel data input registers for SI0 port.
SI1_L/SI1_R	Left/right channel data input registers for SI1 port.
SO0_L/SO0_R	Left/right channel data output registers for SO0 port.
SO1_L/SO1_R	Left/right channel data output registers for SO1 port.

Table 3–12. Serial Port Control Bits

Name	Description
CNS	Suppress right channel output.
DIRI/DIRO	LSB or MSB first select for input/output data.
FI/FO	Standard or I^2S input/output format select.
PBCI/PBCO	BCKI/BCKO active edge select.
PLRI/PLRO	LRCKI/LRCKO sub–frame definition select.
SIM/SOM	Input/output data format select.

Table 3–13. Serial I/O Syntax

Name	Description
SI registers	The four serial input registers : SI0_L, SI0_R, SI1_L, SI1_R
SO registers	The four serial output registers : SO0_L, SO0_R, SO1_L, SO1_R

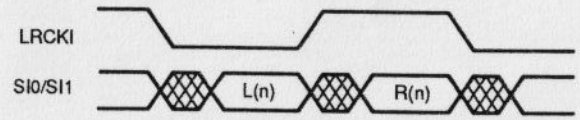
3.8.1 Serial Input Port

The architecture of the serial input port is illustrated in Figure 3–32. The two serial input ports (SI0 and SI1) are controlled by a common LRCKI clock (left/right channel input clock), BCKI (input bit clock) and the contents of status register 0 (ST0).

During one period of LRCKI, two sub–frames of audio data are received. During the first sub–frame, one word of left channel data is received. During the second sub–frame, one word of right channel data is received. These two sub–frames will be referred to as the left and right sub–frames throughout this explanation.

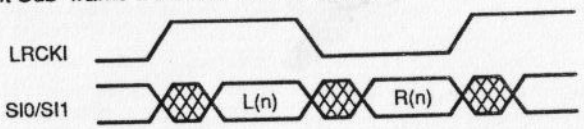
The timing relationship between LRCKI and the left and right sub–frames is dependent on the value of the PLRI control bit. When PLRI=0, LRCKI is low during the left sub–frame and LRCKI is high during the right sub–frame as shown in Figure 3–34.

Figure 3–34. Left/Right Sub–frame Definition when PLRI=0



When PLRI=1, the reverse is true as shown in Figure 3–35.

Figure 3–35. Left/Right Sub–frame Definition when PLRI=1



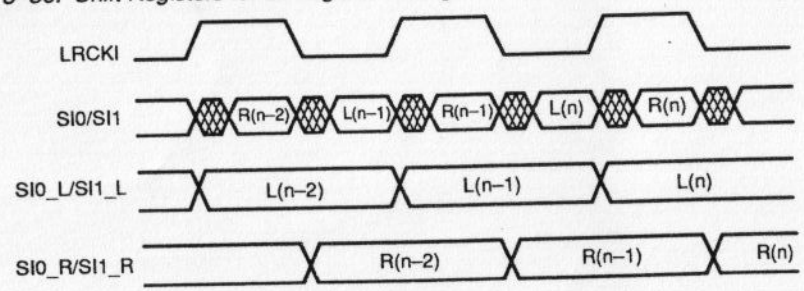
BCKI shifts the audio data from the two input ports into shift registers. Data received from the SI0 port is shifted into SHIFTIN_0 and data received from the SI1 port is shifted into SHIFTIN_1. The serial data are received in the format determined by the settings of the following control bits in the ST0:

- DIRI : LSB or MSB first
- FI : Standard or I^2S format
- SIM : 16 or 24–bit word

For more details on the control bits discussed above please refer to Section 3.8.3.

For both SI0 and SI1 ports, the shift registers are double–buffered to a pair of left and right channel registers. SHIFTIN_0 is double buffered to SI0_L and SI0_R; SHIFTIN_1 is double buffered to SI1_L and SI1_R. The transfer of the shift register contents to the SI registers at the LRCKI clock transition is shown in Figure 3–36.

Figure 3–36. Shift Registers for SI Register Timing



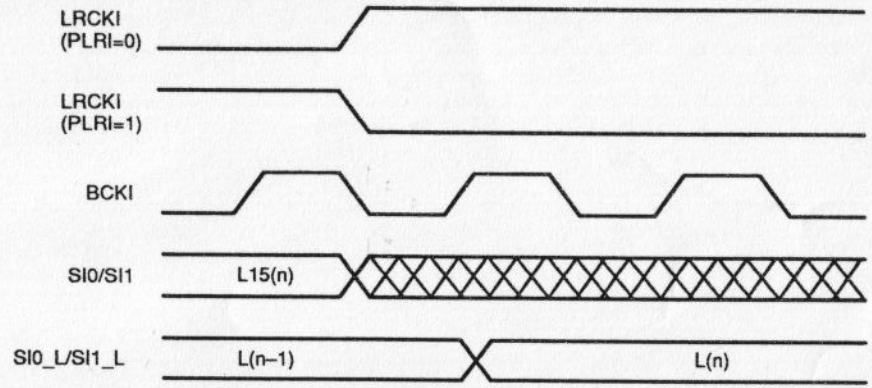
Conditions : PLRI=0, FI=0

Note: The timing for left channel registers (SI0_L and SI1_L) and right channel registers (SI0_R and SI1_R) is different.

After the transition of the LRCKI clock, the new data in the SI registers become valid after a one half BCKI clock delay as shown in Figure 3–37 and

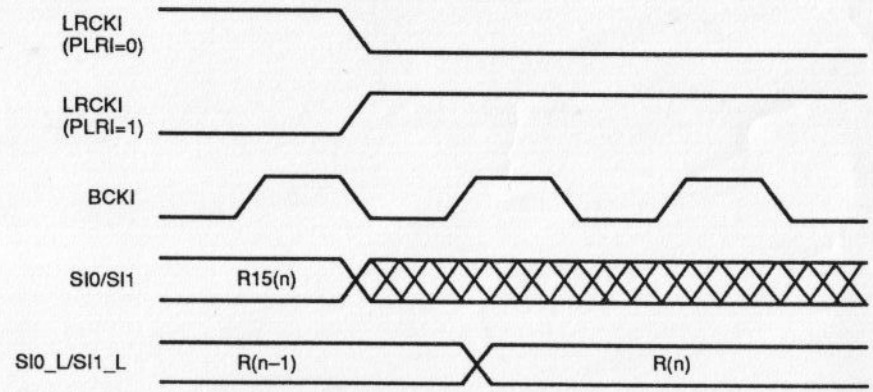
Figure 3–38. Because data are changing at this point, be careful to avoid incorrect readings.

Figure 3–37. Serial Input Timing for Left Channel Data



Conditions : PBCI=0, DIRI=0, FI=0

Figure 3–38. Serial Input Timing for Right Channel Data



Conditions : PBCI=0, DIRI=0, FI=0

The contents of the SI registers can accessed with one of the following three instructions:

- DIS : Loads an SI register value to DMEM.
- DIMH : Loads an SI register value to high MACC.
- DIML : Loads an SI register value to low MACC.

Please refer to the DIS, DIMH and DIML instruction descriptions for details in Section 4.5.

Note:

In order to avoid reading the previously sampled data or the following data sample, instruction execution timing and serial input port timing must be coordinated correctly. For information on instruction execution timing, please refer to Section 3.6.2, Pipeline Operation and Section 3.6.6, Synchronization.

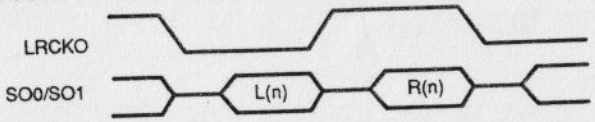
3.8.2 Serial Output Port

The architecture of the serial output port is illustrated in Figure 3–33. The two serial output ports (SO0 and SO1) are controlled by a common LRCKO clock (left and right channel output clock), BCKO (output bit clock) and the contents of status register 0 (ST0).

During one period of LRCKO, two sub–frames of audio data are transmitted to ports SO0 and SO1. During the first sub–frame, one word of left channel data is transmitted. During the second sub–frame, one word of right channel data is transmitted. These two sub–frames will be referred to as the left and right sub–frames throughout this explanation.

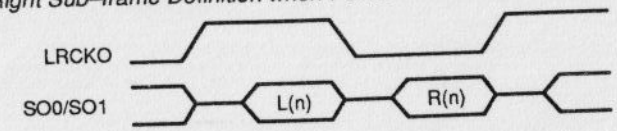
The timing relationship between LRCKO and the left and right sub–frames is dependent on the value of the PLRO control bit. When PLRO=0, LRCKO is low during the left sub–frame and LRCKO is high during the right sub–frame as shown in Figure 3–39.

Figure 3–39. Left/Right Sub–frame Definition when PLRO=0



When PLRO=1, the reverse is true as shown in Figure 3–40.

Figure 3–40. Left/Right Sub–frame Definition when PLRO=1



Similar to the serial input port architecture, each output port has a left and right channel data register double–buffered to a shift register. For the SO0 port,

SO0_L (left register) and SO0_R (right register) are double buffered to SHIFTOUT_0. For the SO1 port, SO1_L and SO1_R are double buffered to SHIFTOUT_1.

The BCKO signal shifts the data from SHIFTOUT_0/SHIFTOUT_1 to SO0/SO1. The active edge of BCKO is determined by the PBCO control bit in the ST0. The format of the transmitted audio data is determined by the settings of the following control bits :

- CNS : Suppress right channel output at SO0 and SO1
- DIRO : LSB or MSB first
- FO : Standard or I^2S format
- SOM : 16 or 24–bit word, ratio of BCKO/LRCKO select

For more details please refer to Section 3.8.3, Serial Port Control Bit Descriptions.

Audio data can be loaded to the SO registers with the following instructions :

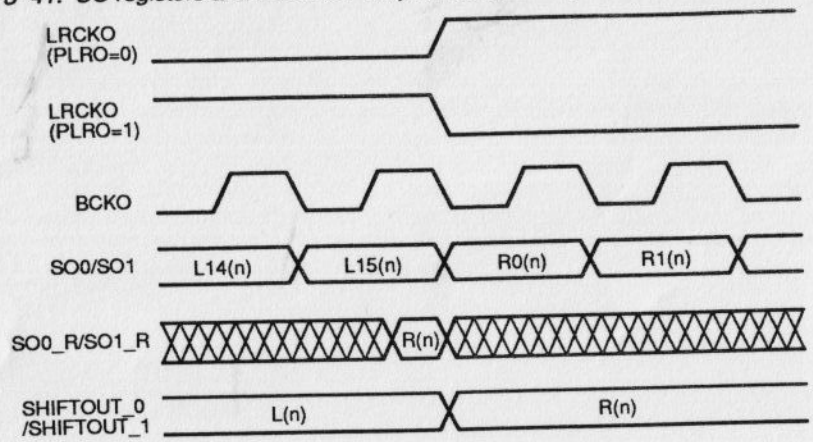
- DOS : Loads a DMEM word to an SO register.
- DOMH : Loads the high MACC value to an SO register.
- DOML : Loads a low MACC value to an SO register.

Please refer to the DOS, DOMH and DOML instruction descriptions in Section 4.5 for details.

The timing of these DOS, DOMH and DOML instructions is critical. At the rising and falling edges of LRCKO, an SO register value is transferred to a shift register (SHIFTOUT_0/SHIFTOUT_1). Therefore, the SO registers must be loaded before the triggering edge of LRCKO.

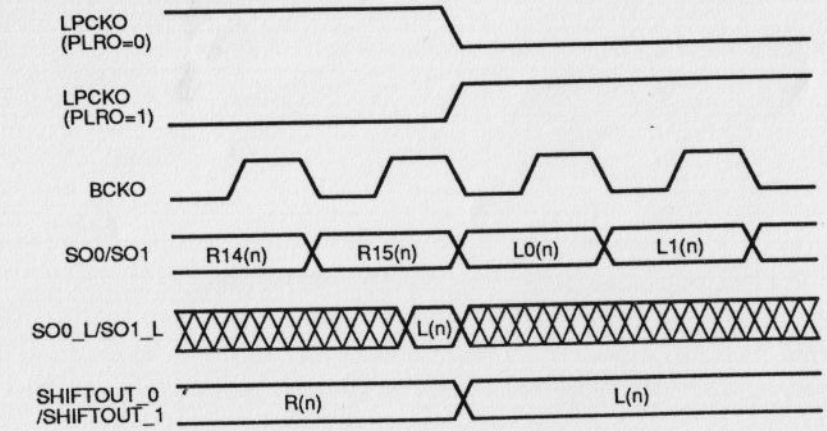
In Figure 3–41 and Figure 3–42, the timing between the SO registers, the shift registers and SO0/SO1 output is illustrated when PLRO=0 and PLRO=1 for left and right channel data.

Figure 3–41. SO registers and SO0/SO1 Output Timing for Right Channel Data



Conditions : DIRO=0, FO=0, PBCO=0, SOM=3

Figure 3–42. SO registers and SO0/SO1 Output Timing for Left Channel Data



Conditions : DIRO=0, FO=0, PBCO=0, SOM=3

Note: In order to avoid transferring the previously sampled data or the following data sample, instruction execution timing and serial output port timing must be coordinated correctly. For information on instruction execution timing, please refer to Section 3.6.2, Pipeline Operation and Section 3.6.6, Synchronization.

3.8.3 Serial Port Control Bit Descriptions

This section describes in detail all control bits in the ST0 that affect the serial port.

3.8.3.1 CNS (Suppress Right Channel Output)

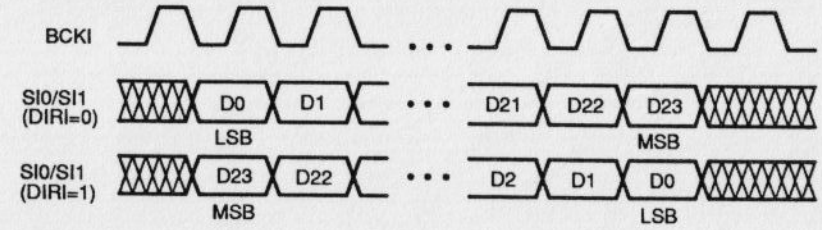
CNS controls whether or not right channel output is suppressed. If CNS=0, both left and right channel data are outputted at SO0 and SO1. If CNS=1, the right channel serial output is suppressed(HI–Z) at both the SO0 and SO1 ports. $\overline{RS}$ resets CNS to 0.

The CNS control bit is used for applications in which left and right channel data must be ORed. For example, assume that there are two 5700X devices and each one is dedicated to only one channel of data. One device is used for left channel data and the second is used for right channel data. Set CNS to one in both devices. Because the right channel data will be suppressed in both devices, in the second device, use the left channel timing sequence to output the right channel data. By setting PLRO to zero in the first device and PLRO to one in the second device the data from the first and second device will be outputted out of phase. This has effectively ORed the left and right channel data together.

3.8.3.2 DIRI (Receive Data LSB or MSB First)

DIRI controls whether data is received LSB or MSB first. If DIRI=0, the two serial input ports (SI0 and SI1) will receive input audio data LSB first. If DIRI=1, the data will be received MSB first. $\overline{RS}$ resets DIRI to 0. See Figure 3–43.

Figure 3–43. Input Timing when DIRI=0 and DIRI=1

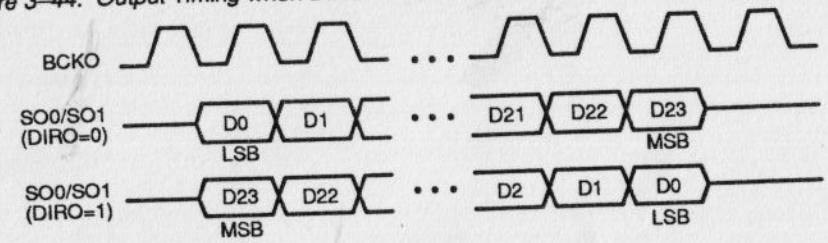


Conditions : PBCI=0, SIM=0

3.8.3.3 DIRO (Transmit Data LSB or MSB First)

DIRO controls whether data is transmitted LSB or MSB first. If DIRO=0, the two serial output ports (SO0 and SO1) will transmit audio data LSB first. If DIRO=1, the data will be transmitted MSB first. $\overline{RS}$ resets DIRO to 0. See Figure 3–44.

Figure 3–44. Output Timing when DIRO=0 and DIRO=1

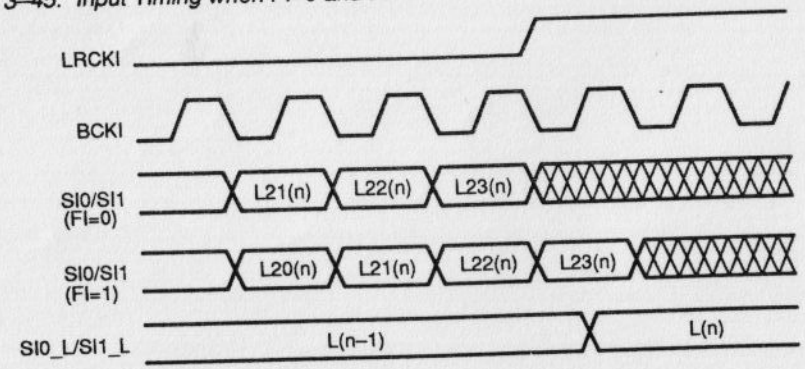


Conditions : PBCO=0, SOM=0

3.8.3.4 FI (Standard or I²S Format for Input Data)

FI sets the serial input timing to that of standard format or I²S format. If FI=0, standard format is selected. The last data bit is received before the triggering edge of LRCKI. If FI=1, I²S format is selected. The last data bit is received after the triggering edge of LRCKI. With both standard and I²S format, the data in the shift register is latched to an SI register by the triggering edge of BCKI immediately following the triggering edge of LRCKI. $\overline{RS}$ resets FI to 0. Please see Figure 3–45.

Figure 3–45. Input Timing when FI=0 and FI=1



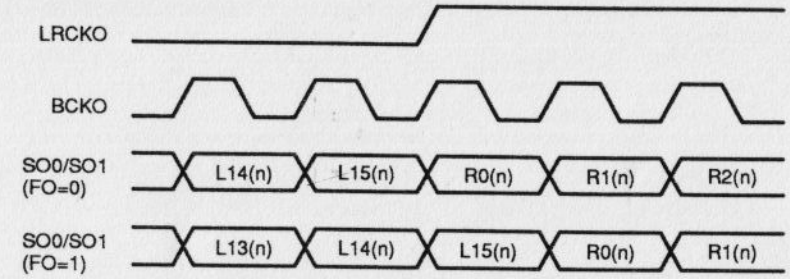
Conditions : DIRI=0, PBCI=0, PLRI=0, SIM=0

3.8.3.5 FO (Standard or I²S Format for Output Data)

FO sets the serial output timing to that of standard format or I²S format. If FO=0, standard format is selected. The last data bit is transmitted before the

triggering edge of LRCKO. If FO=1, I²S format is selected. The last data bit is transmitted after the trigger edge of LRCKO. $\overline{RS}$ resets FO to 0. Please see Figure 3–46.

Figure 3–46. Output Timing when FO=0 and FO=1

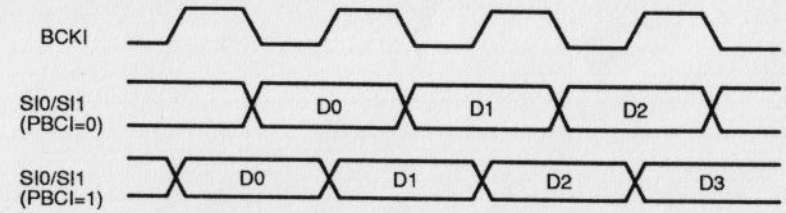


Conditions : DIRO=0, PBCO=1, PLRO=0, SOM=3

3.8.3.6 PBCI (BCKI Active Edge Select)

PBCI controls the active edge of BCKI. If PBCI=0, at the rising edge of BCKI, audio data from the serial input ports (SI0 and SI1) are shifted into the shift registers (SHIFTIN_0 and SHIFTIN_1). If PBCI=1, at the falling edge of BCKI, audio data are shifted into the shift registers. $\overline{RS}$ resets PBCI to 0. Please see Figure 3–47.

Figure 3–47. Input Timing when PBCI=0 and PBCI=1



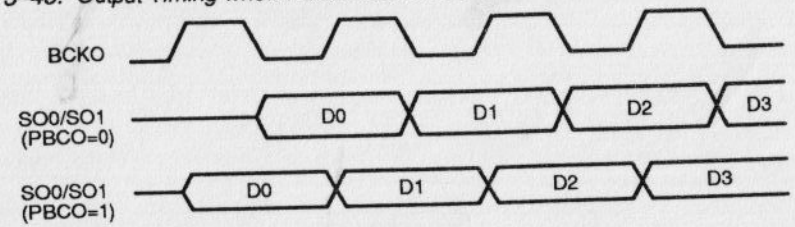
Condition : DIRI=0

3.8.3.7 PBCO (BCKO Active Edge Select)

PBCO controls the active edge of BCKO. If PBCO=0, at the falling edge of BCKO, the output shift register contents (SHIFTOUT_0 and SHIFTOUT_1) are shifted to the serial output ports (SO0 and SO1). If PBCO=1, at the rising

edge of BCKO, the data are shifted out. $\overline{RS}$ resets PBCO to 0. Please see Figure 3–48.

Figure 3–48. Output Timing when PBCO=0 and PBCO=1

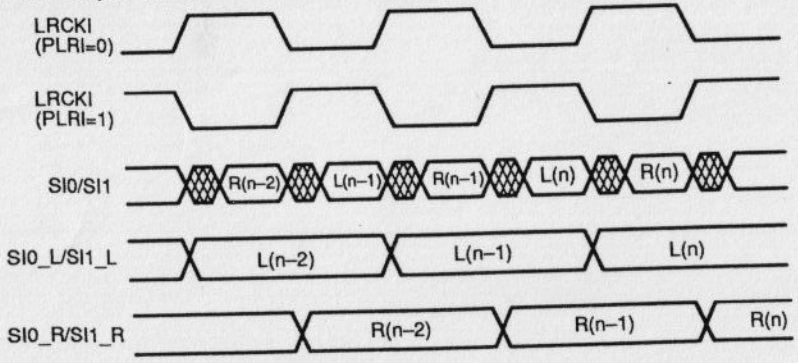


Conditions : DIRO=0, SOM≠3

3.8.3.8 PLRI (LRCKI Sub–frame Definition Select)

PLRI controls the sub–frame definition of LRCKI with respect to the input of one word of left and right channel data. If PLRI=0, the left channel data are received when LRCKI is low and the right channel data are received when LRCKI is high. If PLRI=1, the left channel data are received when LRCKI is high and the right channel data are received when LRCKI is low. $\overline{RS}$ resets PLRI to 0. Please see Figure 3–49.

Figure 3–49. Output Timing when PLRI=0 and PLRI=1



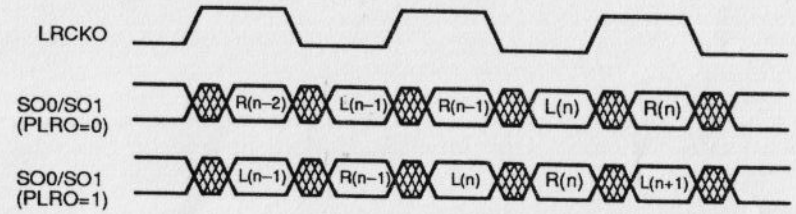
Condition : FI=0

3.8.3.9 PLRO (LRCKO Sub–frame Definition Select)

PLRO controls the sub–frame definition of LRCKO with respect to the output of one word of left and right channel data. If PLRO=0, the left channel data are

transmitted when LRCKO is low, and the right channel data are transmitted when LRCKO is high. If PLRO=1, the left channel data are transmitted when LRCKO is high, and the right channel data are transmitted when LRCKO is low. $\overline{RS}$ resets PLRI to 0. Please see Figure 3–50.

Figure 3–50. Output Timing when PLRO=0 and PLRO=1

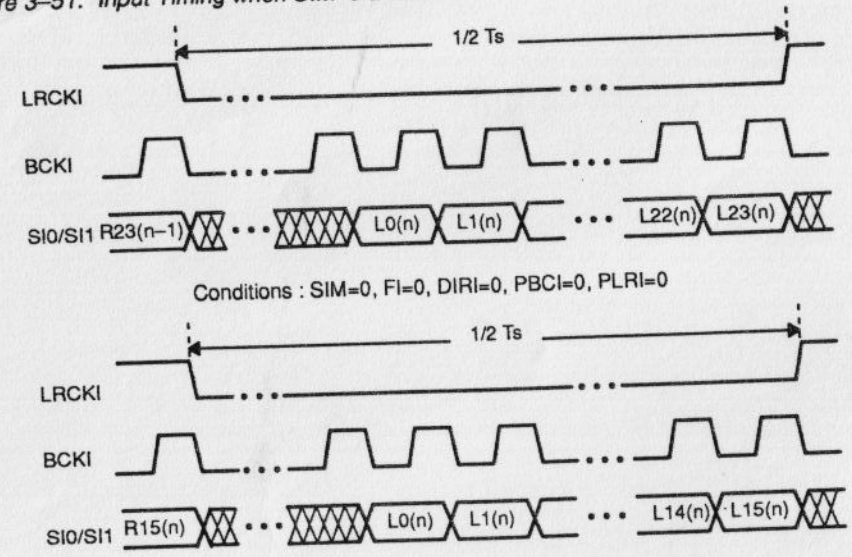


Condition : FO=0

3.8.3.10 SIM (Input Audio Data Format)

SIM controls the input audio data format. If SIM=0, the data word length is 24 bits for both input ports (SI0 and SI1). If SIM=1, the data word length is 16 bits. For a 16–bit data word, a 24–bit data value is created by extending the LSBs with 8 zero–filled bits. $\overline{RS}$ resets SIM to 0. Please see Figure 3–51.

Figure 3–51. Input Timing when SIM=0 and SIM=1



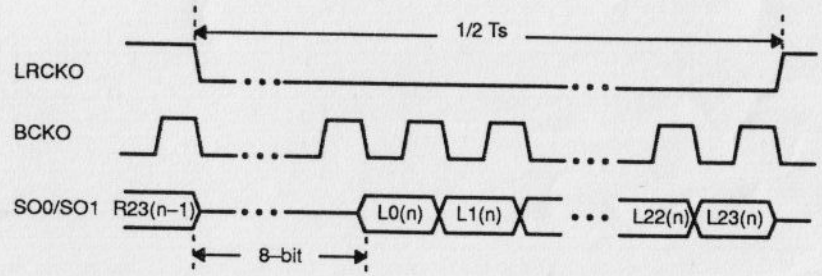
3.8.3.11 SOM (Output Audio Data Format)

SOM controls the output audio data format. The audio data format is determined by the data word length and the ratio of BCKO/LRCKO. The different formats available are summarized in Table 3–14. $\overline{RS}$ resets SOM to 0. Refer to Figure 3–52 and Figure 3–53 for timing diagrams.

Table 3–14. Output Data Format Summary

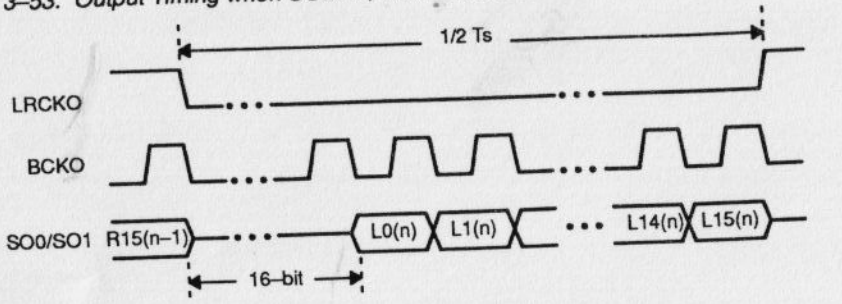
Data Format (SOM)	SOM=0	SOM=1	SOM=2	SOM=3
Word Bit Length	24	16	16	16
BCKO/LRCKO	64	64	48	32

Figure 3–52. Output Timing when SOM=0

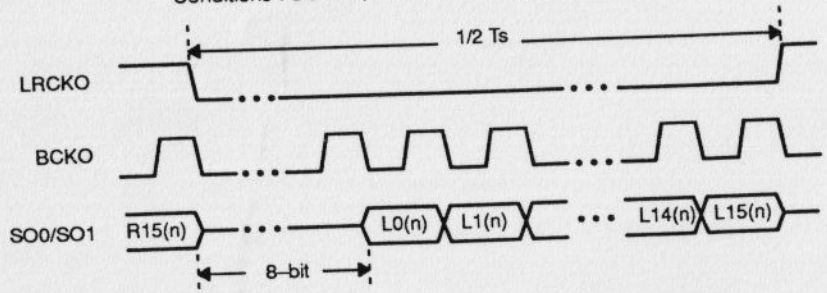


Conditions : SOM=0, FO=0, DIRO=0, PBCO=0, PLRO=0

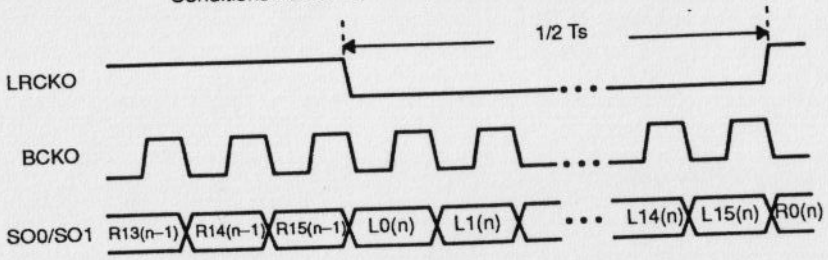
Figure 3–53. Output Timing when SOM=1, SOM=2 and SOM=3



Conditions : SOM=1, FO=0, DIRO=0, PBCO=0, PLRO=0



Conditions : SOM=2, FO=0, DIRO=0, PBCO=0, PLRO=0



Conditions : SOM=3, FO=0, DIRO=0, PBCO=0, PLRO=0

3.9 External RAM Interface (TMS57002 only)

The architecture of the TMS57002 includes an external RAM interface. With the external RAM interface, the architecture of the TMC57000/1 has been expanded to provide flexibility for the implementation of audio applications that require handling delayed data.

The main features of the external RAM interface are listed below:

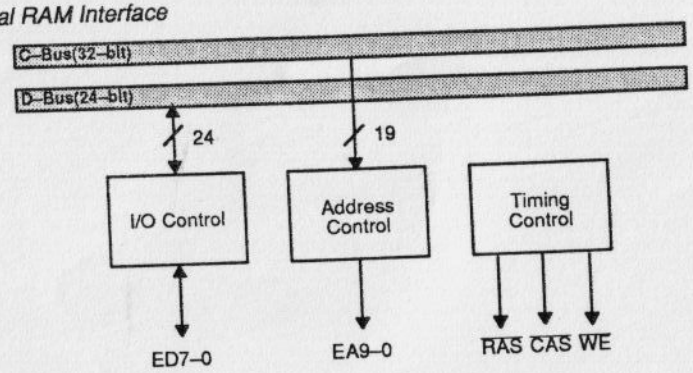
- ❑ 256K to 8M–bit external RAM
- ❑ 16 or 24–bit data word length
- ❑ 4 or 8–bit I/O port width
- ❑ DRAM or SRAM capability
- ❑ External memory used as ring buffer for audio data
- ❑ External RAM access instructions
  - RDE, SRBD : read access
  - WRE : write access
  - REF : refresh

In the case of maximum external RAM capability (8M–bit), the maximum delay time is approximately 10 seconds for 16–bit data at 48KHz sampling.

3.9.1 External RAM Interface Configuration

The external RAM interface is configured as shown in Figure 3–54 below.

Figure 3–54. External RAM Interface



The external RAM interface contains three major components: I/O Control, Address Control and Timing Control. The I/O Control handles data transfers between external memory and data memory (DMEM). Refer to Section 3.9.3 for more on I/O Control. The Address Control, calculates the address used to access external memory. To calculate this access address, coefficient memory (CMEM) must be accessed because it contains the offset address into external memory. Refer to Section 3.9.5 for more on Address Control. The Timing Control creates the $\overline{RAS}$, $\overline{CAS}$ and $\overline{WE}$ signals from the master clock.

3.9.2 Status Register Control of the External RAM Interface

The external RAM interface is controlled by four control bits in ST0. These bits are summarized below in Table 3–15.

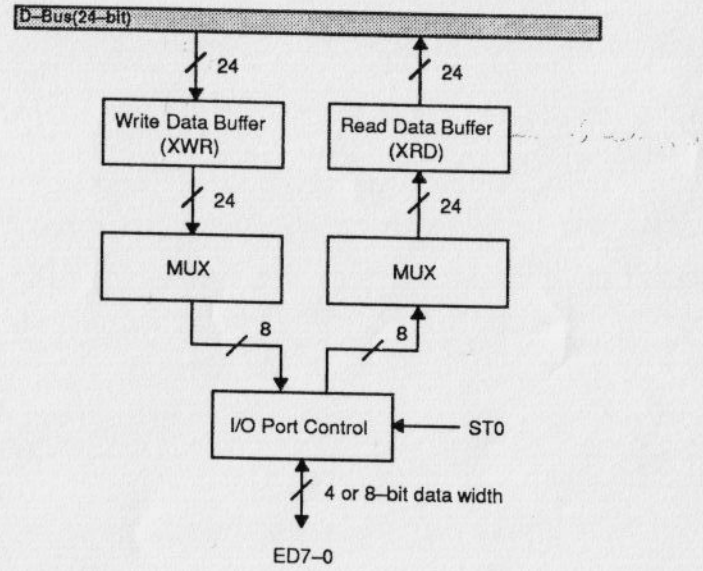
Table 3–15. External RAM Interface Control Bits

Name	Bit Number	Description
M	17, 16	M=0 : 64K address space M=1 : 256K address space M=2 : 1M address space
SEL	15	SEL=0 : 4–bit I/O port width SEL=1 : 8–bit I/O port width
SRAM	21	SRAM=0 : DRAM used as external RAM SRAM=1 : SRAM used as external RAM
WORD	14	WORD=0 : 16–bit data WORD=1 : 24–bit data

3.9.3 I/O Control

The I/O Control is illustrated in Figure 3–55 below. The external RAM interface supports 16 or 24–bit data (see WORD control bit) and a 4 or 8–bit I/O port (see SEL control bit). Select these options by setting the required control bits as described in Section 3.9.2.

Figure 3–55. I/O Control



The number of fetches required to access external memory is dependent upon the length of the data word and the bit–width of the I/O port. As illustrated in Table 3–16 below there are four possible cases:

Table 3–16. Fetches Required for External RAM Access

Audio Data Width	I/O Port Width	# of Fetches
16–bit	4–bit	4
16–bit	8–bit	2
24–bit	4–bit	6
24–bit	8–bit	3

The WRE instruction writes one data memory(DMEM) word to external memory via the XWR buffer (external memory write data buffer). The RDE instruction reads one word of data from external memory and stores it in the XRD

buffer (external memory read data buffer). The SRBD instruction loads the contents of the XRD buffer to DMEM.

3.9.4 External RAM as a Circular Buffer

The external RAM is implemented as a single circular buffer. The implementation of external RAM as a circular buffer is similar to that of the implementation of DMEM as a circular buffer. At synchronization, the base address register, XBA is decremented by one. Because the value of XBA points to the memory location of the oldest data sample (assuming that the offset address register, XOA is set to zero), the oldest data sample in memory can be replaced by the data sample just received.

To access a data sample k samples before the newest data sample, simply set the XOA register to k. The offset address (k) is stored in the lower 19–bits of coefficient memory (CMEM). This address is loaded to the XOA register via the C–Bus during the execution of the WRE and RDE instructions.

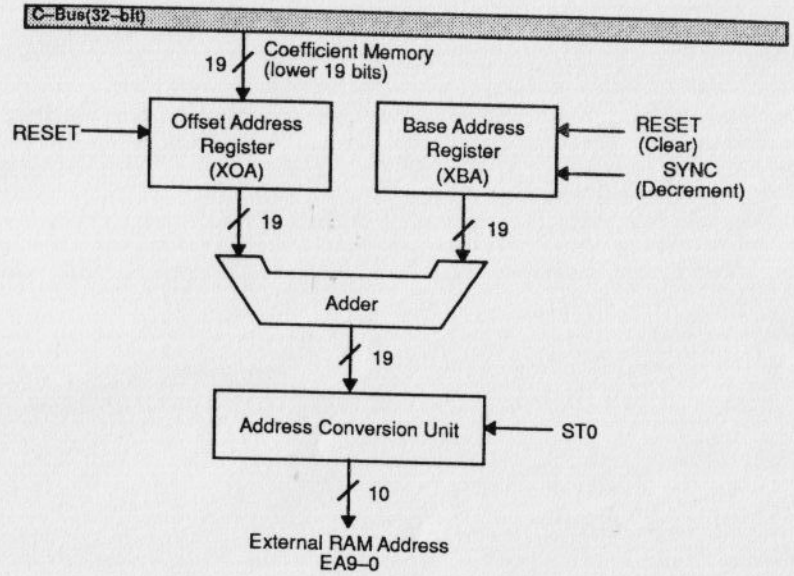
With external RAM, it is possible to implement multiple delay buffers. A common application is to divide memory into two, one half for right channel data and one half for left channel data. Please refer to Section 5.3 for more details on implementing delay buffers.

3.9.5 Address Control

The external RAM interface supports memory with addressing capability of 64K, 256K and 1M. Select from among these options by setting the M control bits as described in Section 3.9.2. The Address Control is illustrated in Figure 3–56.

At synchronization, the base address register (XBA), is decremented by one. The contents of the offset address register (XOA) are stored in the lower 19 bits of CMEM. CMEM can be updated with the host controller (see Section 3.7.3). XOA and XBA are added in the Adder. The result passes to the Address Conversion Unit where the address is outputted to the EA9 through EA0 pins as summarized in Table 3–17.

Figure 3–56. Address Control



In order to implement page mode access and reduce the need for refresh operations, the access address into external memory has been assigned to pins as illustrated in Table 3–17. The access address into external memory is determined by concatenating the higher bits of the sum of XBA and XOA (represented by ax) to the number of fetches required to access one word of data (represented by Cx). As illustrated in Figure 3–56, XBA and XOA are added in the Adder. In the Address Conversion Unit, the higher bits of the sum of XBA and XOA are extended by Cx. The address is then outputted in a time shared fashion to the EA9 through EA0 pins as summarized in Table 3–17.

Table 3–17. Addressing Pin Assignment

			EA9	EA8	EA7	EA6	EA5	EA4	EA3	EA2	EA1	EA0
64K	ROW		0	0	a0	a7	a6	a5	a4	a3	a2	a1
	COL	2 Fetches	0	0	a14	a13	a12	a11	a10	a9	a8	C0
		3 Fetches	0	0	a13	a12	a11	a10	a9	a8	C1	C0
		4 Fetches	0	0	a13	a12	a11	a10	a9	a8	C1	C0
		6 Fetches	0	0	a12	a11	a10	a9	a8	C2	C1	C0
256K	ROW		0	a0	a8	a7	a6	a5	a4	a3	a2	a1
	COL	2 Fetches	0	a16	a15	a14	a13	a12	a11	a10	a9	C0
		3 Fetches	0	a15	a14	a13	a12	a11	a10	a9	C1	C0
		4 Fetches	0	a15	a14	a13	a12	a11	a10	a9	C1	C0
		6 Fetches	0	a14	a13	a12	a11	a10	a9	C2	C1	C0
1M	ROW		a9	a0	a8	a7	a6	a5	a4	a3	a2	a1
	COL	2 Fetches	a18	a17	a16	a15	a14	a13	a12	a11	a10	C0
		3 Fetches	a17	a16	a15	a14	a13	a12	a11	a10	C1	C0
		4 Fetches	a17	a16	a15	a14	a13	a12	a11	a10	C1	C0
		6 Fetches	a16	a15	a14	a13	a12	a11	a10	C2	C1	C0

In the case of 64K (see Figure 3–57), to represent the row and column addresses, 16 bits are required. The bit–length of the sum of XBA and XOA is equal to:

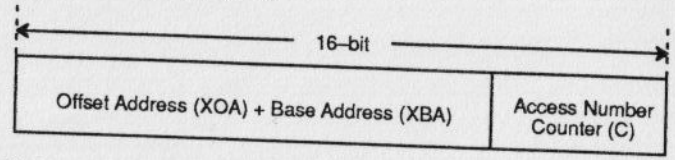
16 – (length of C field)

The higher bits of the sum of XBA and XOA are discarded.

The number of bits in the C field is determined by the number of fetches required to access one word of data (see Table 3–16). The C field is used as a counter to access one word of data the required number of times. For example, for 16–bit audio data with a 4–bit I/O port, four fetches are required for read/write access. Therefore, a two bit counter (C1, C0) fulfills this requirement (see Figure 3–57).

The size of the C field allows for the support of page mode access. For example, in the case of 24–bit audio data and an 8–bit I/O port, three fetches are required to access one word. As can be seen in Figure 3–57 in Case 3, the C field is two bits. Because two bits allows for four fetches, one fourth of memory is not used. This prevents words from being accessed across row boundaries, thereby increasing preformance.

Figure 3–57. Addressing Pin Assignment (64K)



CASE 1. 16–bit Audio Data, 8–bit I/O Port $0 \le C \le 1$

Address:	a_{14}	a_{13}	a_{12}	•	•	•	a_2	a_1	a_0	C_0
	EA9	EA8	EA7	EA6	EA5	EA4	EA3	EA2	EA1	EA0
ROW:	0	0	a_0	a_7	a_6	a_5	a_4	a_3	a_2	a_1
COL:	0	0	a_{14}	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	C_0

CASE 2. 24–bit Audio Data, 8–bit I/O Port $0 \le C \le 2$

Address:	a_{13}	a_{12}	a_{11}	•	•	•	a_1	a_0	C_1	C_0
	EA9	EA8	EA7	EA6	EA5	EA4	EA3	EA2	EA1	EA0
ROW:	0	0	a_0	a_7	a_6	a_5	a_4	a_3	a_2	a_1
COL:	0	0	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	C_1	C_0

CASE 3. 16–bit Audio Data, 4–bit I/O Port $0 \le C \le 3$

Address:	a_{13}	a_{12}	a_{11}	•	•	•	a_1	a_0	C_1	C_0
	EA9	EA8	EA7	EA6	EA5	EA4	EA3	EA2	EA1	EA0
ROW:	0	0	a_0	a_7	a_6	a_5	a_4	a_3	a_2	a_1
COL:	0	0	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	C_1	C_0

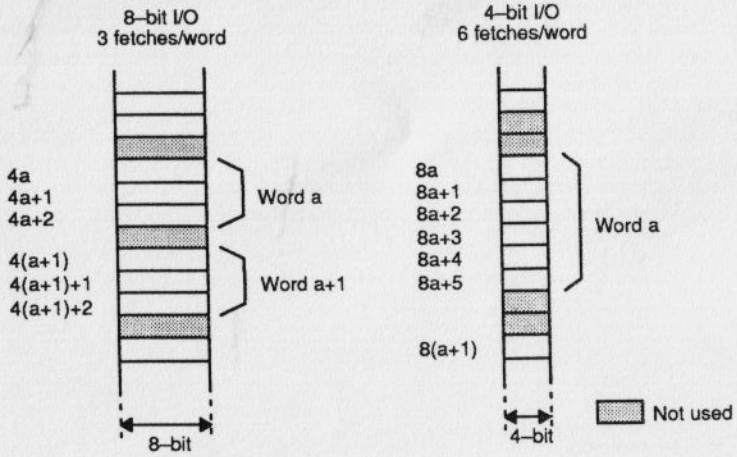
CASE 4. 24–bit Audio Data, 4–bit I/O Port $0 \le C \le 5$

Address:	a_{12}	a_{11}	a_{10}	•	•	•	a_0	C_2	C_1	C_0
	EA9	EA8	EA7	EA6	EA5	EA4	EA3	EA2	EA1	EA0
ROW:	0	0	a_0	a_7	a_6	a_5	a_4	a_3	a_2	a_1
COL:	0	0	a_{12}	a_{11}	a_{10}	a_9	a_8	C_2	C_1	C_0

As illustrated in Figure 3–58, in order to implement page mode access, in the case of 24–bit data, with an 8–bit I/O port, every fourth address location is un-

assigned; with a 4–bit I/O port, every seventh and eighth memory location are unassigned.

Figure 3–58. 24–bit Word Memory Allocation



Because the address bits are not assigned to EA9 through EA0 in order, the need for a refresh cycle has been eliminated in most cases. To fulfill the requirement that memory must be refreshed within 4ms (256K), in the slowest case at 32KHz sampling, all rows must be accessed once within a 128 synchronization cycle (128 = 4ms/1/32KHz). Since there are 256 rows, two rows must be accessed per synchronization cycle.

The row address and column address are outputted in reverse order at the EA9 through EA0 pins. Therefore, adjacent words in memory will have different row addresses. In the worst case, (ie. a program that writes a data value to memory at the present base address and reads the previous data value) 129 rows will be accessed. With most applications, 256 rows will be accessed within 4ms, eliminating the need for refresh. The REF instruction is provided for applications requiring refresh capability. Please see Section 3.9.8.

3.9.6 External RAM Access Summary

The external RAM interface supports both DRAM and SRAM. Select one of these two options by setting the SRAM control bit (see Section 3.9.2). The features and access timing for DRAM and SRAM are summarized in Table 3–18 and Table 3–19, respectively.

Table 3–18. External DRAM Summary (48 kHz sampling cycle)

DRAM Density		256K				1M				4M	
DRAM Type		256K × 1		64K × 4		1M × 1		256K × 4		1M × 4	
Number of DRAM		8	4	2	1	8	4	2	1	2	1
Address Space (Number of Bits)		256K (18)	256K (18)	64K (16)	64K (16)	1M (20)	1M (20)	256K (18)	256K (18)	1M (20)	1M (20)
Number of Audio Data Words	24 bit/word	64K	32K	16K	8K	256K	128K	64K	32K	256K	128K
	16 bit/word	128K	64K	32K	16K	512K	256K	128K	64K	512K	256K
MAX Delay Time (ms)	24 bit/word	1365	682	341	170	5461	2730	1365	682	5461	2730
	16 bit/word	2730	1365	682	341	10922	5461	2730	1365	10922	5461
Access Time (machine cycles)	24 bit/word	8	14	8	14	8	14	8	14	8	14
	16 bit/word	6	10	6	10	6	10	6	10	6	10
Number of Access (word/sampling)	24 bit/word	32	18	32	18	32	18	32	18	32	18
	16 bit/word	42	25	42	25	42	25	42	25	42	25

Table 3–19. External SRAM Summary

Data I/O width (bit)		8
Number of Audio Data Words	24 bit/word	(SRAM Density)/8/4
	16 bit/word	(SRAM Density)/8/2
Max Delay Time (ms)	24 bit/word	(# of Audio Data Words) × (Sampling Time)
	16 bit/word	(# of Audio Data Words) × (Sampling Time)
Access Time (machine cycles)	24 bit/word	15
	16 bit/word	11
Number of Access (word/sampling)	24 bit/word	17
	16 bit/word	23

3.9.7 Read and Write Access (DRAM and SRAM)

The RDE instruction loads the 19 lower bits from CMEM into the XOA register (offset address register) via the C–Bus, and then read access to external RAM is triggered. With the WRE instruction, one word of data from DMEM is loaded to the XWR buffer (write data buffer) via the D–Bus. In parallel, the WRE instruction loads the 19 lower bits from CMEM into XOA, and then the WRE

instruction triggers write access to external RAM. With both the RDE and WRE instructions, the sum of XOA and XBA is used in determining the access address into external RAM (see Section 3.9.5). The address is outputted at pins E9 through E0 as summarized in Table 3–17.

The RDE and WRE instructions require one machine cycle to trigger read and write access. It takes additional cycles to complete this processing. The total required number of machine cycles is determined by data length (16 or 24–bit), the I/O port width (4 or 8–bit) and the RAM type (DRAM or SRAM). Refer to Section 3.9.2 on how to set the required control bits.

In the case of DRAM, the number of machine cycles required to complete RDE and WRE processing is equal to 2(m+1), m being the number of fetches to access one word (see Table 3–16). For SRAM, a total of 11 machine cycles are required for 16–bit data and 15 machine cycles are required for 24–bit data. For a summary of the required machine cycles for DRAM and SRAM refer to Table 3–18 and Table 3–19, respectively. If the WRE, RDE or REF instructions are executed during the required machine cycle time needed for RDE or WRE processing, they will be ignored.

The SRBD instruction loads the contents of the XRD buffer to an addressed location in DMEM. If the SRBD instruction is executed during the required machine cycle time needed for RDE processing, the XRD buffer will contain the value from the previous processing of the RDE instruction.

DRAM read and write access timing are illustrated in Figure 3–59 and Figure 3–61. Page mode access allows for continuous read and write access.

SRAM read and write access timing are illustrated in Figure 3–60 and Figure 3–62. An external row address latch is required since there is no time sharing of the row/column addresses at the E9 through E0 pins. With SRAM, only an 8–bit I/O port is supported.

Figure 3–59. Read Access Timing for External DRAM (CLKSEL="H")

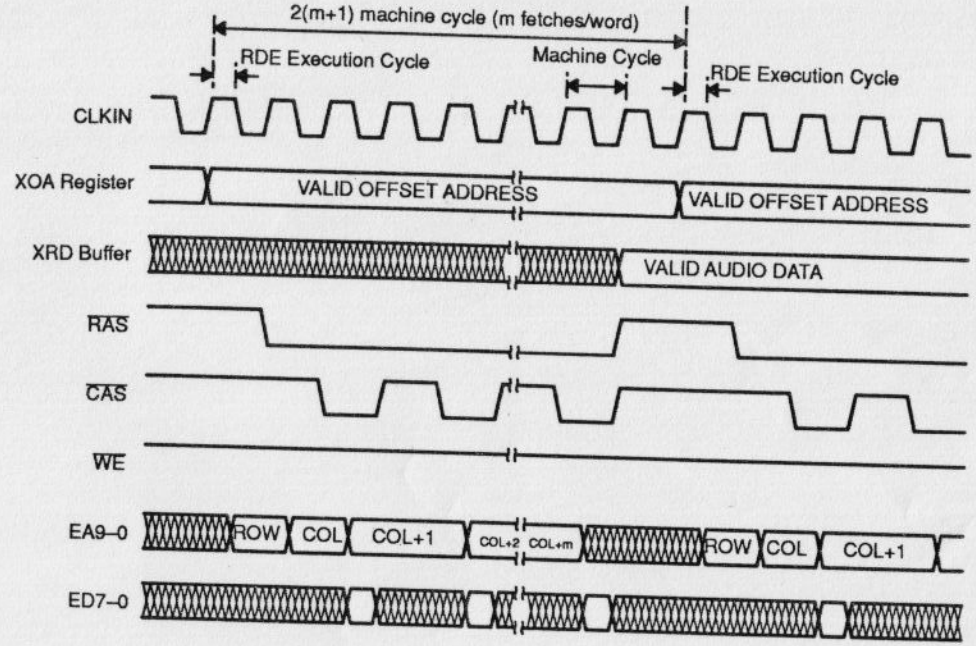


Figure 3–60. Read Access Timing for External SRAM(CLKSEL="H") for 16–bit Audio Data

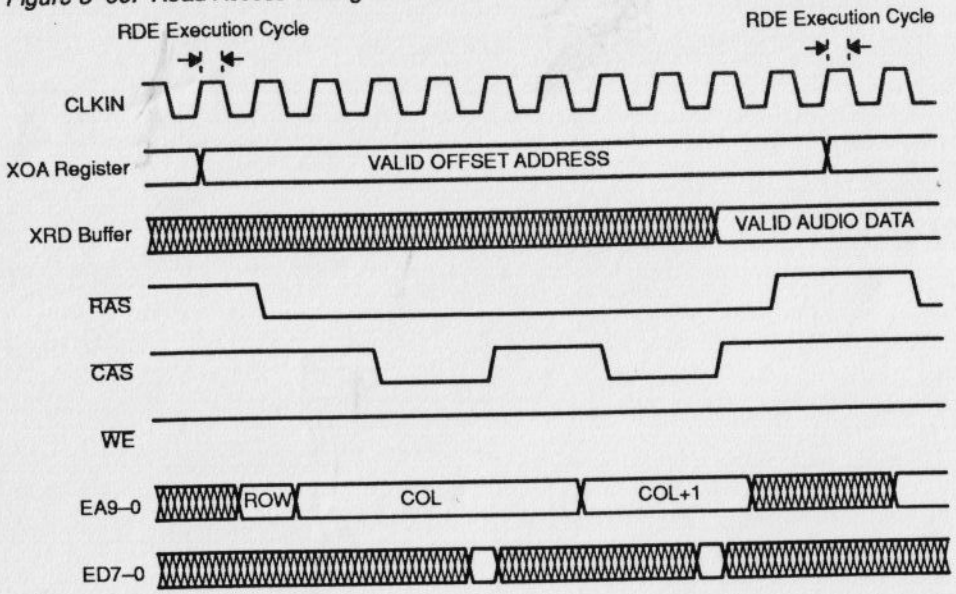


Figure 3–61. Write Access Timing for External DRAM (CLKSEL = "H")

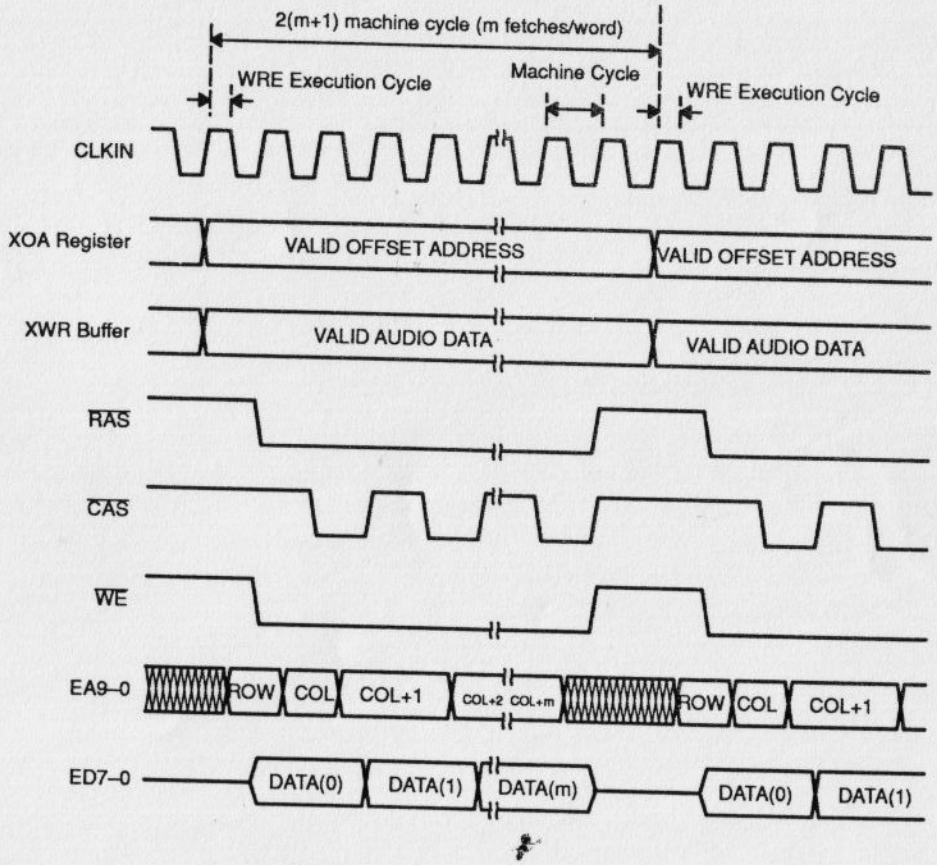
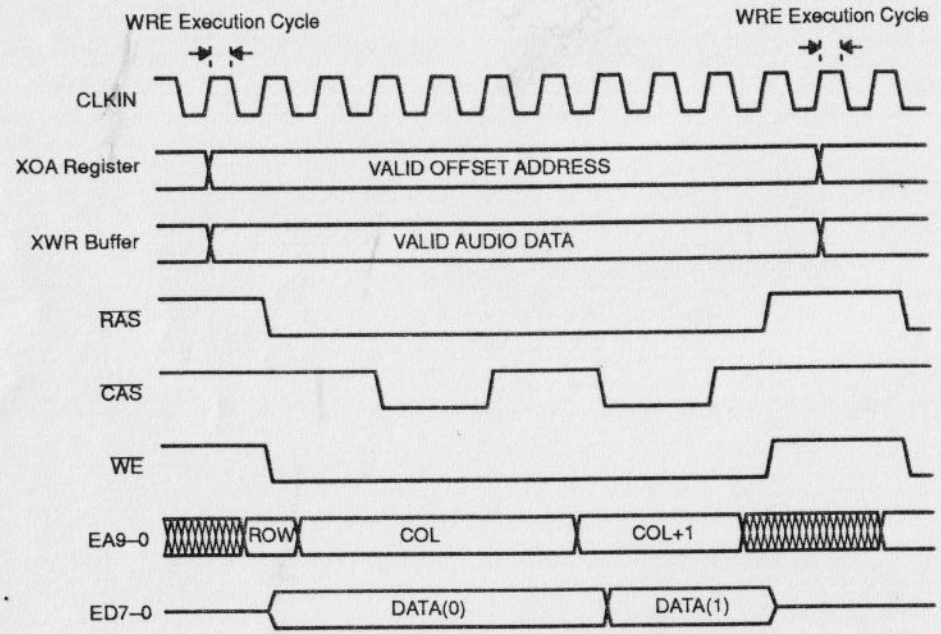


Figure 3–62. Write Access Timing for External SRAM (CLKSEL="H") for 16–bit Audio Data

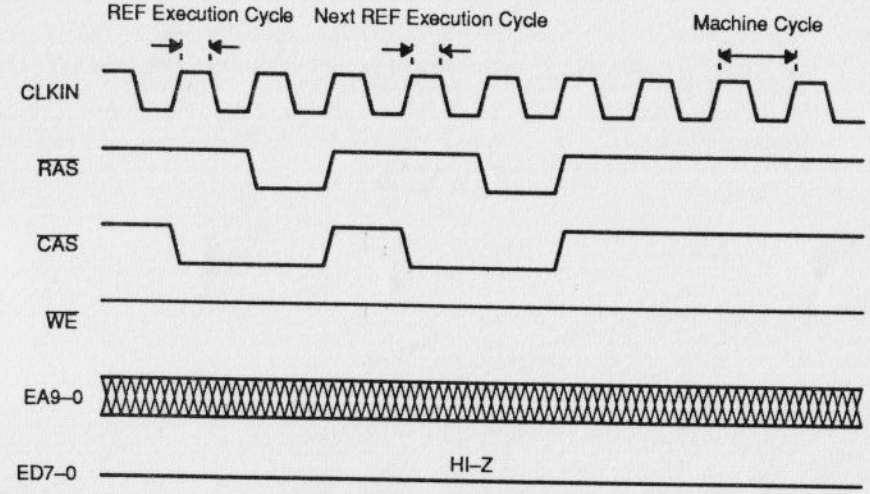


3.9.8 Refresh Timing

When DRAM is used for external memory, a refresh cycle must be performed every 4 ms (256K) or every 8 ms (1MB) to retain data. To satisfy refresh operation conditions, as explained previously in Section 3.9.5, most applications will access all rows within the required refresh time, eliminating the need for refresh. Yet, for applications that do not satisfy this condition, the REF instruction is available.

The REF instruction can be used at any time. It requires one machine cycle to trigger refresh execution and two additional machine cycles to complete this processing. Therefore, the two instructions following an REF instruction should not include an RDE, WRE or REF instruction. If one of these instructions is executed during the timing required for REF processing, the REF instruction takes precedence and the newly executed instruction will be ignored. Please refer to Figure 3–63 for DRAM refresh timing.

Figure 3–63. Refresh Timing for External DRAM (CLKSEL="H")



Chapter 4

Assembly Language Instructions

The TMC57000, TMC57001 and TMS57002 instruction sets support audio signal processing operations, including linear phase FIR filter operations. In addition, the TMS57002, with its external memory interface, supports applications handling delayed data. To support these external memory interface operations in addition to supporting a bi–directional host interface which is unique to the TMS57002, the TMS57002 instruction set includes six additional instructions. These instructions are included as part of the instruction set descriptions in this chapter but are clearly marked as only available for the TMS57002.

This chapter describes the assembly language instructions for the TMC57000, TMC57001 and TMS57002. Each of the individual instructions is described and listed in alphabetical order. An example instruction demonstrates the special format used and explains its content.

Topic Page

4.1	Parallel Execution of Instructions	4-2
4.2	Instruction Word Definition	4-4
4.3	Memory Addressing Modes	4-6
4.4	Instruction Set	4-13
4.5	Individual Instruction Descriptions	4-21

4.1 Parallel Execution of Instructions

Parallel execution of instructions is possible with the 5700x Series. Two instructions can be executed in a single cycle which greatly reduces total program line number. For example, the following code,

```
DIS 0, D*
MPY D*+, C*+
```

can be reduced to one line as follows:

```
MPY D*+, C*+ / DIS 0
```

In the line above, the first instruction will be referred to as the primary instruction and the secondary instruction will be referred to as the secondary instruction. The primary and secondary instructions must be separated by a '/'.

In general, primary instructions perform calculations and secondary instructions transfer data. When a primary and secondary instruction are executed in parallel, a bus conflict is avoided because the secondary instruction will take precedence.

All instructions belong to one of three categories: Category 1 (primary instructions), Category 2 (secondary instructions) and Category 3 (branch or immediate instructions). Branch and immediate instructions can not be executed in parallel with another instruction. The different categories are summarized in Table 4–1 below:

Table 4–1. Instruction Set Categorization by Operational Function

Category	Instruction Types
Category 1 (Primary Instruction)	ALU operation MAC operation
Category 2 (Secondary Instruction)	ALU data transfer MAC data transfer Coefficient memory data transfer Data input/output operation Status register modification
Category 3 (Branch/Immediate Instruction)	Immediate load operation Branch operation

In addition to categorization by a number, primary and secondary instructions are further categorized by a letter based upon bus usage and memory access of the instruction as summarized in Table 4–2.

Table 4–2. Instruction Set Categorization by Memory Access and Bus Usage

Memory/bus Access	Primary Instruction	Secondary Instruction
Data memory read only	A	A
Data memory write only	B	B
Coefficient memory read only	C	C
Coefficient memory write only	D	D
Data and coefficient memory read	E	–
Data memory read and coefficient memory write	F	–
Data memory write and coefficient memory read	G	–
Data memory write and coefficient memory write	H	–
ALU accumulator or MAC accumulator to B–Bus transfer	I	–
Data transfer using D–Bus	–	J
No Bus usage	N	N

The valid and invalid combinations of primary and secondary instructions are summarized in Table 4–3. Valid combinations are designated by a 'v' and invalid combinations are designated by a hyphen. The assembler will detect invalid combinations and give a warning. Parallel execution of two primary or two secondary instructions is also invalid.

Table 4–3. Valid Primary and Secondary Instruction Combinations

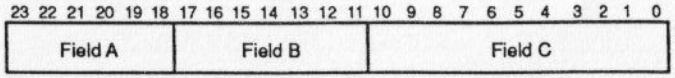
		Primary Instruction (Category 1)									
		A	B	C	D	E	F	G	H	I	N
Secondary Instruction (Category 2)	A	v	v	v	v	v	v	v	v	v	v
	B	v	–	v	v	v	v	–	–	v	v
	C	v	v	v	v	v	v	v	v	v	v
	D	v	v	v	–	v	–	–	–	v	v
	J	–	–	v	v	–	–	–	–	v	v
	N	v	v	v	v	v	v	v	v	v	v

key:"v"=valid combination, "–"=invalid combination

4.2 Instruction Word Definition

The instruction word is a 24–bit value. The 24–bit instruction word is divided into three fields, A, B and C as illustrated in Figure 4–1 below:

Figure 4–1. Instruction Word Fields



Field A is the primary instruction (Category 1) field. The function of field B and field C is defined by the value of field A as illustrated in Table 4–4 below:

Table 4–4. Field Definition

Field A	Field B	Field C
A ≠ 3Fh	Category 2 instruction (Secondary instruction)	Memory address
A=3Fh	Category 3 instruction (Immediate/branch instruction)	Immediate value/PC value

When field A is not equal to 3Fh, field A contains the primary instruction, field B contains the secondary instruction, and field C contains the memory address information. The memory addressing field is described further in the following section (Section 4.2.1).

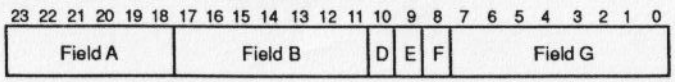
When field A is equal to 3Fh, the instruction word contains an immediate or branch instruction (Category 3 instruction). Category 3 instruction fields are described further in Section 4.2.2.

4.2.1 Memory Addressing Field

The TMC57000, TMC57001 and TMS57002 architecture can handle two memory accesses in any one instruction line. (The AMAC and AMPY instructions are the only exception in which there are three memory accesses.) This includes all memory accesses in both the primary and secondary operation instruction fields. For purposes of explanation, these two memory accesses will be referred to as Memory X and Memory Y. Memory X can be a direct or indirect memory access. Memory Y can be only an indirect access. The assembler controls the assignment of the memory accesses to Memory X and Memory Y.

When an instruction accesses memory, field C (bits 10 through 0) is further divided into fields D, E, F and G as illustrated in Figure 4–2.

Figure 4–2. Instruction Field Allocation for Memory Addressing



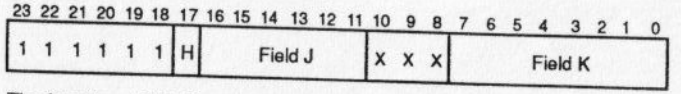
These fields are used to describe the type of memory access: data or coefficient memory , direct or indirect addressing, post increment or no post increment. The information pertaining to Memory X is contained in field D, field F and field G. The information pertaining to Memory Y is contained in field D and field E. The fields are defined as follows:

1) Field D:Coefficient/Data Memory Select
   If D=0, Memory X is a data memory access.
   Memory Y is a coefficient memory access.
   If D=1, Memory X is a coefficient memory access.
   Memory Y is a data memory access.
2) Field E:Memory Y Post Increment Addressing Select
   If E=0, no post increment.
   If E=1, post increment.
3) Field F:Memory X Direct/ Indirect Addressing Select
   If F=0, indirect addressing.
   If F=1, direct addressing with the 8–bit G field value.
4) Field G:Memory X Addressing Field Definition
   (Case 1) Memory X has indirect addressing (F=0).
   Only bit 7 of field G is used:
   If bit 7=0, no post increment.
   If bit 7=1, post increment.
   (Case 2) Memory X has direct addressing (F=1).
   Field G contains the direct addressing value.
   This value is added to the base address register value to calculate the actual memory address.

4.2.2 Category 3 (Immediate/Branch) Instruction Fields

In the case of Category 3 instructions, field B is further divided into field H and field J. In field C, only bits 7 through 0 are valid (field K). See Figure 4–3.

Figure 4–3. Field Allocation for Category 3 Instructions



The function of field J and field K is defined by the value of field H as illustrated in Table 4–5:

Table 4–5. Field Allocation for Category 3 Instructions

Field H	Field J	Field K
0	Immediate instruction	Immediate value
1	Branch instruction	PC value

4.3 Memory Addressing Modes

The TMC57000, TMC57001 and TMS57002 instruction sets provide three memory addressing modes:

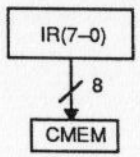
- Direct addressing
- Indirect addressing
- Immediate addressing

Both direct and indirect addressing can be used to access coefficient and data memory. Included in the following sections is a description of each addressing mode complete with sample opcode and examples. Because of the differences in architecture between data and coefficient memory addressing, they are treated separately.

4.3.1 Direct Addressing

With direct addressing, the 8 lower bits of the instruction word contain either the coefficient memory address or the data memory address. In the case of coefficient memory, the coefficient memory address is the actual address used to access memory. This value is stored in the IR register. See Figure 4–4 below.

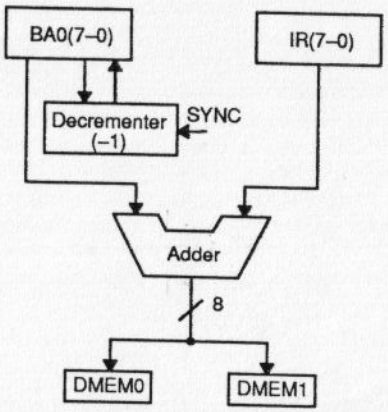
Figure 4–4. Coefficient Memory Direct Addressing



In the case of data memory, the access address into data memory is the sum of the DMEM0 base address register, BA0 and IR. Although the coefficient memory direct addressing architecture is the same for all devices, the data memory direct addressing architecture differs slightly between the TMC57000, TMC57001 and TMS57002.

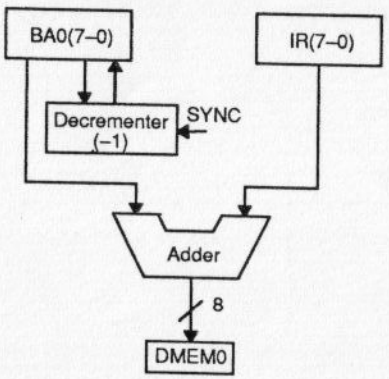
In the case of the TMC57000, direct addressing can be used to access both DMEM0 and DMEM1. DMEM0 and DMEM1 are accessed as memory pages. The DBP control bit points to one of the two data memory banks and the DMEM0 base address register, BA0, is used to calculate the memory address for access into both DMEM0 and DMEM1. At synchronization, BA0 is decremented by one. See Figure 4–5.

Figure 4–5. TMC57000 and TMS57002 (CAS=0) Data Memory Direct Addressing



In the case of the TMC57001, only DMEM0 can be accessed with direct addressing. The 8–bit data memory access address is formed identically to that of the TMC57000, except that DMEM1 can not be accessed. Figure 4–6 illustrates the direct addressing of the TMC57001:

Figure 4–6. TMC57001 and TMS57002 (CAS=1) Data Memory Direct Addressing



In the case of the TMS57002, the direct addressing format is controlled by the CAS control bit. When CAS=0, the TMS57002 direct addressing format is the same as that of the TMC57000. When CAS=1, the TMS57002 direct addressing format is the same as that of the TMC57001.

The following symbols are used in direct addressing:

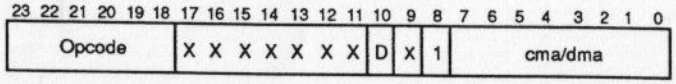
C(K) Direct addressing for coefficient memory (CMEM). The contents of IR(7–0) are used as the coefficient memory address.

D(K) Direct addressing for data memory (DMEM). The contents of BA0(7–0) plus IR(7–0) are used as the data memory address.

K Constant (0≤K≤255)

Direct addressing is available for all instructions in Category 1 and the following instructions in Category 2: DIS, DOS, LD0T, LPC(TMS57002 only), LPD(TMS57002 only), SACC, SACD, SLML, SMHC, SMHD, SMLD. Within an instruction line, there can be only one direct addressing memory reference. As illustrated below, the direct addressing instruction format is the same for Category 1 and Category 2 instructions except for the opcode location.

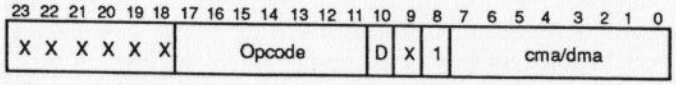
Figure 4–7. Direct Addressing Format (Category 1)



Bits 23 through 18 contain the primary instruction opcode. The addressing mode is defined as direct (bit 8 =1). Field D is the memory select field. When field D=0, data memory is accessed with direct addressing and bits 7 through 0 contain the data memory address (dma). When field D=1, coefficient memory is accessed and bits 7 through 0 contain the coefficient memory address (cma). Refer to Section 4.2 for more on the fields of an instruction word.

The direct addressing format is as follows for Category 2 instructions:

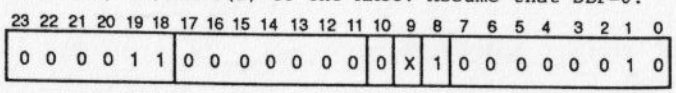
Figure 4–8. Direct Addressing Format (Category 2)



Bits 17 through 11 contain the secondary instruction opcode. The remaining fields are identical to Category 1 instruction fields (see above explanation). An example of direct addressing is provided in Example 4–1.

Example 4–1. Direct Addressing Format

```
ADD D(2), A ;Add D(2) to the AACC. Assume that DBP=0.
```

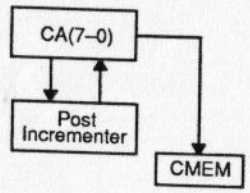


The primary instruction opcode of the above example is contained in bits 23 through 18. Since a secondary operation instruction is not specified, bits 17 through 11 are zero–filled. (The opcode of the default secondary instruction, NOP2, is zero.) Data memory is accessed (bit 10 = 0) and the addressing mode is defined as direct (bit 8 = 1). Bits 7 through 0 contain the data memory offset address value, 2h. This value is added with BA0, the base address register for DMEM0, to calculate the data memory access address.

4.3.2 Indirect Addressing

Both coefficient and data memory can be accessed with indirect addressing. In the case of coefficient memory (CMEM), the current value of the CA register is used to access CMEM. In the case of post increment, CA will be incremented by one after the memory address is calculated. See Figure 4–9 below.

Figure 4–9. Coefficient Memory Indirect Addressing



Indirect addressing can also be used to access both DMEM0 and DMEM1. The access address into data memory is the sum of the current value of the index register (ID) and the contents of the base address register (BA0 or BA1).

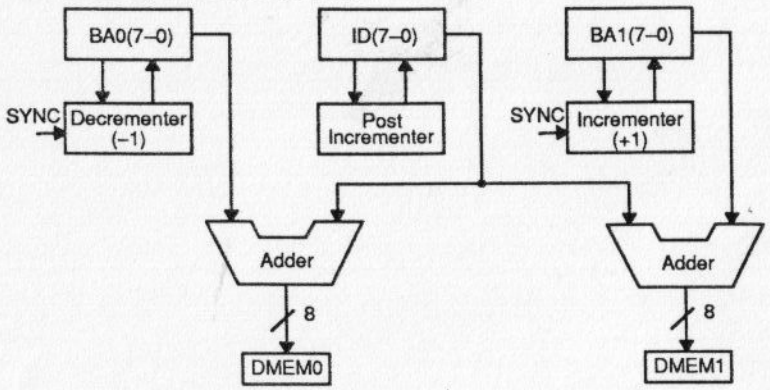
The base address register used is dependent upon the 5700x device. In the case of the TMC57000 and TMS57002 (CAS=0), for indirect addressing of both DMEM0 and DMEM1, BA0 is used as the base address register. In the case of the TMC57001 and TMS57002 (CAS=1), if DMEM0 is selected (DBP=0), the base address register is BA0. If DMEM1 is selected (DBP=1), the base address register is BA1.

Note:

For all 5700x devices, the ADDS, AMPY, AMAC, LD0T and STD1 instructions are an exception. With these instructions, the value of DBP is ignored. For indirect addressing of DMEM0, BA0 is used as the base address register. For indirect addressing of DMEM1, BA1 is used as the base address register.

In the case of post increment, the ID register will be incremented by one after the DMEM address is calculated. Figure 4–10 illustrates how the 8–bit data memory address is formed.

Figure 4–10. Data Memory Indirect Addressing



The following symbols are using in indirect addressing:

C* The contents of CA(7–0) are used as the coefficient memory address.

C*+ The contents of CA(7–0) are used as the coefficient memory address and incremented after the access.

D* The contents of BA0 or BA1(DMEM1 of TMC57001 and TMS57002, CAS=1, only) plus ID(7–0) are used as the data memory address.

D*+ The contents of BA0 or BA1(DMEM1 of TMC57001 and TMS57002, CAS=1, only) plus ID(7–0) are used as the data memory address, and ID(7–0) is incremented after the access.

Indirect addressing can be used with all instructions in Category 1 and the following instructions in Category 2: DIS, DOS, LD0T, STD1, LPC(TMS57002 only), LPD(TMS57002 only), SACC, SACD, SLML, SMHC, SMHD, SMLD. Within an instruction line, there can be up to two indirect memory accesses. For purposes of explanation, these two memory accesses are referred to as Memory X and Memory Y. Refer to Section 4.2.1 for more details.

In the case of two indirect addressing memory accesses, the assignment to Memory X and Memory Y is assembler controlled. This is also true in the case of an instruction line containing only one indirect addressing memory access. In the case of two memory accesses, one of direct addressing and the other of indirect addressing, the direct addressing memory access is assigned to Memory X, and the indirect addressing mode memory access is assigned to Memory Y.

The instruction word format for an indirect addressing memory access assigned to Memory Y is illustrated in Figure 4–11.

Figure 4–11. Memory Y Indirect Addressing Format

23 22 21 20 19 18	17 16 15 14 13 12 11	10	9	8	7 6 5 4 3 2 1 0
Primary Instruction Opcode	Secondary Instruction Opcode	D	E	X	X X X X X X X X

Field E above is the post increment field (E=0, no post increment; E=1, post increment) and field D is the memory select field (D=0, data memory access; D=1, coefficient memory access).

The instruction word format for an indirect addressing memory access assigned to Memory X is illustrated in Figure 4–12.

Figure 4–12. Memory X Indirect Addressing Format

23 22 21 20 19 18	17 16 15 14 13 12 11	10	9	8	7	6 5 4 3 2 1 0
Primary Instruction Opcode	Secondary Instruction Opcode	D	X	0	P	X X X X X X X

Memory X can have either direct or indirect addressing. As shown above, bit 8 =0, defines Memory X as having indirect addressing. Field D is the memory select field (D=0, data memory access; D=1, coefficient memory access) and field P (bit 7) is the post increment field (P=0, no post increment; P=1, post increment). An example of indirect addressing is provided in Example 4–2.

Example 4–2. Indirect/Direct Addressing Format

```
MPY C*, D(2) ;Multiply the coefficient memory value and
             ;data memory value.
```

23 22 21 20 19 18	17 16 15 14 13 12 11	10	9	8	7 6 5 4 3 2 1 0
1 0 0 0 0 1	0 0 0 0 0 0 0	0	0	1	0 0 0 0 0 0 1 0

The primary instruction opcode of Example 4–2 is contained in bits 23 through 18. Since a secondary operation instruction is not specified, bits 17 through 11 are zero–filled. (The opcode of the default secondary instruction, NOP2, is zero.)

The direct addressing access, D(2), is assigned to Memory X. The addressing mode is defined as direct (bit 8 =1). Bits 7 through 0 contain the data memory offset address value, 2h. Memory X is defined as a data memory access (bit 10=0). The indirect addressing access, C*, is assigned to Memory Y. Memory Y is defined as a coefficient memory access (bit 10 = 0) with no post increment (bit 9 = 0). The contents of the CA register are used as the coefficient memory address.

4.3.3 Immediate Addressing

Included in the TMC57000, TMC57001 and TMS57002 instruction sets are five immediate operand instructions, in which the immediate operand is con-

tained within the instruction word. The bit–length of the immediate operand is instruction–dependent. The immediate instructions are:

LBRK
Load DMEM0 base address register (BA0) immediate (8–bit constant)

LCAK
Load coefficient address register (CA) immediate (8–bit constant)

LDPK
Load data memory bank pointer (DBP) immediate (1–bit constant)

LIRK
Load data memory index register (ID) immediate (8–bit constant)

RPTK
Repeat instruction as specified by immediate value (8–bit constant)

The instruction word format depends upon whether the immediate instruction is a Category 2 or Category 3 instruction as illustrated by the following examples.

Example 4–3. Immediate Addressing Format (Category 3)

```
RPTK 99   ;Execute the instruction following this
          ;instruction 100 times.
```

The instruction format for RPTK is as follows:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	0	0	1	0	0	0	0	X	X	X	Field K							

The immediate operand is contained in field K. This is true for all Category 3 immediate addressing mode instructions (LBRK, LCAK and LIRK).

Example 4–4. Immediate Addressing Format (Category 2)

```
LDPK 0 ;Load the data memory bank pointer (DBP) with 0.
```

The instruction format for LDPK is as follows:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	1	0	0	0	1	0	0	X	X	X	X	X	X	X	X	X	X	X

LDPK is the only immediate addressing mode instruction in Category 2. The 1–bit immediate operand is specified in bit 11.

4.4 Instruction Set

The instruction set includes the following:

- ❑ Symbols and Abbreviations
- ❑ Instruction Set Summary
- ❑ Instruction Bus Usage

4.4.1 Symbols and Abbreviations

Table 4–6 lists the symbols and abbreviations used in the instruction set summary (Table 4–7) and the individual instruction descriptions (Section 4.5).

Table 4–6. Instruction Symbols and Abbreviations

Symbol	Description
AACC cma constant dma dma0 dma1 DMEM0 DMEM1 MACC PC pma	ALU (arithmetic logic unit) accumulator Coefficient memory address Immediate operand value Data memory address (DMEM0 or DMEM1) DMEM0 (Data memory bank 0) memory address DMEM1 (Data memory bank1) memory address Data memory bank 0 Data memory bank 1 MAC (Multiplier Adder) accumulator Program counter Program memory address
k x	Constant bit field for immediate operand Don't care value
X AND Y X OR Y X XOR Y	Bitwise logical–AND of X and Y Bitwise logical–OR of X and Y Bitwise logical–XOR of X and Y
mapping(X) →Y sign(X) unsigned(X)	Y is loaded with value specified by X The sign bit of the value X The unsigned value of X
X → Y | | < > [ ] ( ) { } /	X is assigned to Y Absolute value User–defined items Optional items Contents of Alternative items; one of which must be entered Separates between items in a list

4.4.2 Instruction Set Summary

The instruction set summary of the TMC57000, TMC57001 and TMS57002 is listed in Table 4–7. The instructions are arranged according to function and alphabetized within each functional grouping. A detailed description of each instruction is provided in Section 4.5.

There are six instructions which are included only in the TMS57002 instruction set. They are marked clearly as being available only for the TMS57002. All other instructions are available for the TMC57000, TMC57001 and TMS57002.

Table 4–7. Instruction Set Summary

Mnemonic	Description	Category	Operation
ALU Accumulator Memory Reference Instructions			
ABS	Absolute value of ALU accumulator	1I	|(AACC)|→AACC
ADD(1)	Add data memory to ALU accumulator	1A	(dma)+(AACC)→AACC
ADD(2)	Add coefficient memory to ALU accumulator	1C	(cma)+(AACC)→AACC
ADD(3)	Add data memory and MAC accumulator	1A	(dma)+(MACC)→AACC
ADD(4)	Add coefficient memory and MAC accumulator	1C	(cma)+(MACC)→AACC
ADD(5)	Add data memory and coefficient memory	1E	(dma)+(cma)→AACC
ADDS	Add right–shifted data memory bank 0 and right–shifted data memory bank 1	1A	$(dma0)\times2^{-1}+(dma1)\times2^{-1}\rightarrow AACC$
AND(1)	AND data memory with ALU accumulator	1A	(dma) AND (AACC)→AACC
AND(2)	AND coefficient memory with ALU accumulator	1C	(cma) AND (AACC)→AACC
AND(5)	AND data memory with coefficient memory	1E	(dma) AND (cma)→AACC
CMPL	Complement ALU accumulator	1I	$\overline{(AACC)}\rightarrow AACC$
LACC	Load ALU accumulator with coefficient memory	1C	(cma)→AACC
LACD	Load ALU accumulator with data memory	1A	(dma)→AACC
NEG	Negate ALU accumulator	1I	(AACC)×(−1)→AACC
OR(1)	OR data memory with ALU accumulator	1A	(dma) OR (AACC)→AACC
OR(2)	OR coefficient memory with ALU accumulator	1C	(cma) OR (AACC)→AACC
OR(5)	OR data memory with coefficient memory	1E	(dma) OR (cma)→AACC
SUB(1)	Subtract ALU accumulator from data memory	1A	(dma)–(AACC)→AACC
SUB(2)	Subtract ALU accumulator from coefficient memory	1C	(cma)–(AACC)→AACC
SUB(3)	Subtract MAC accumulator from data memory	1A	(dma)–(MACC)→AACC
SUB(4)	Subtract MAC accumulator from coefficient memory	1C	(cma)–(MACC)→AACC
SUB(5)	Subtract coefficient memory from data memory	1E	(dma)–(cma)→AACC
XOR(1)	Exclusive–OR data memory with ALU accumulator	1A	(dma) XOR (AACC)→AACC
XOR(2)	Exclusive–OR coefficient memory with ALU accumulator	1C	(cma) XOR (AACC)→AACC
XOR(5)	Exclusive–OR data memory with coefficient memory	1E	(dma) XOR (cma)→AACC
ZAC	Zero ALU accumulator	1N	0→AACC
Parallel AACC and MACC Reference Instructions			
AMAC	Multiply coefficient memory by ALU accumulator, accumulate and store in MAC accumulator. In parallel, add shifted data memory banks, store in ALU accumulator.	1E	$(cma)\times(AACC)\times2+(MACC)\rightarrow MACC$ $(dma0)\times2^{-1}+(dma1)\times2^{-1}\rightarrow AACC$
AMPY	Multiply coefficient memory by ALU accumulator, store in MAC accumulator. In parallel, add shifted data memory banks, store in ALU accumulator.	1E	$(cma)\times(AACC)\times2\rightarrow MACC$ $(dma0)\times2^{-1}+(dma1)\times2^{-1}\rightarrow AACC$

Table 4–7. Instruction Set Summary (Continued)

Mnemonic	Description	Category	Operation
MAC Accumulator Memory Reference Instructions			
LMHC	Load high MAC accumulator with sign–extended coefficient memory	1C	$(cma)\rightarrow MACC_{47-16}$, $sign(cma)\rightarrow MACC_{51-48}$, $0\rightarrow MACC_{15-0}$
LMHD	Load high MAC accumulator with sign–extended data memory	1A	$(dma)\rightarrow MACC_{47-24}$, $sign(dma)\rightarrow MACC_{51-48}$, $0\rightarrow MACC_{23-0}$
LMLD	Load low MAC accumulator with data memory	1A	$(dma)\rightarrow MACC_{23-0}$, $MACC_{51-24}\rightarrow MACC_{51-24}$
MAC(1)	Multiply data memory by coefficient memory, accumulate	1E	(dma)×(cma)+(MACC)→MACC
MAC(2)	Multiply ALU accumulator by data memory, accumulate	1A	(AACC)×(dma)+(MACC)→MACC
MAC(3)	Multiply coefficient memory by ALU accumulator, accumulate	1C	(cma)×(AACC)+(MACC)→MACC
MACS	Multiply coefficient memory by left–shifted ALU accumulator, accumulate	1C	(cma)×(AACC)×2+(MACC)→MACC
MACU(1)	Multiply unsigned data memory by coefficient memory, accumulate	1E	unsigned(dma)×(cma)+(MACC) →MACC
MACU(2)	Multiply ALU accumulator by unsigned data memory, accumulate	1A	(AACC)×unsigned(dma)+(MACC) →MACC
MPY(1)	Multiply data memory by coefficient memory	1E	(dma)×(cma)→MACC
MPY(2)	Multiply coefficient memory by ALU accumulator	1C	(cma)×AACC→MACC
MPY(3)	Multiply CREG by data memory	1A	(CREG)×(dma)→MACC
MPYU	Multiply unsigned data memory by coefficient memory	1E	unsigned(dma)×(cma)→MACC
SFML	Shift MAC accumulator 1 bit left	1N	$(MACC)_{49-0}\rightarrow MACC_{50-1}$ $(MACC)_{51}\rightarrow MACC_{51}$ $0\rightarrow MACC_0$
SFMR	Shift MAC accumulator 1 bit right	1N	$(MACC)_{51-1}\rightarrow MACC_{50-0}$ $(MACC)_{51}\rightarrow MACC_{51}$
ZMAC	Zero the MAC accumulator	1N	0→MACC
Auxiliary Registers/Pointers Instructions			
INCD	Decrement DMEM0 base address register, increment DMEM1 base address register	2N	(BA0)−1→BA0 (BA1)+1→BA1
LBRK	Load DMEM0 base address register immediate	3	constant→BA0
LCAA	Load CA register with ALU accumulator	2J	AACC→CA
LCAC	Load CA register conditionally	3	If 0 ≤ AACC then constant→CA else (CA) → CA
LCAK	Load CA register immediate	3	constant→CA
LDOT	Load TREG with data memory bank0	2A	(dma)→TREG
LDPK	Load data memory bank pointer immediate	2N	constant→DBP

Table 4–7. Instruction Set Summary (Continued)

Mnemonic	Description	Category	Operation
	Auxiliary Registers/Pointers Instructions (Continued)		
LIRA	Load ID register with ALU accumulator	2J	(AACC)→ID
LIRK	Load ID register immediate	3	constant→ID
	Branch/Repeat Instructions		
B	Branch unconditionally	3	(pma)→PC
BGZ	Branch if ALU accumulator is greater than zero	3	If (AACC) > 0 then (pma)→PC else (PC)+1→PC
BIOZ	Branch if $\overline{BIO}$ pin is low	3	If ($\overline{BIOZ}$) = 0 then (pma)→PC else (PC)+1→PC
BLZ	Branch if ALU accumulator is less than zero	3	If (AACC) < 0 then (pma)→PC else (PC)+1→PC
BNZ	Branch if ALU accumulator is not equal to zero	3	If (AACC) $\neq$ 0 then (pma)→PC else (PC)+1→PC
BV	Branch on ALU accumulator overflow	3	If (AOV) = 1 then (pma)→PC else (PC)+1→PC
RPTK	Repeat the following instruction one more time than that specified by the immediate value	3	constant→RPTC
	Input/Output Instructions		
DIMH	Load high MAC accumulator with serial input register specified by k.	2J	(SI(k))→$MACC_{47-24}$ sign(SI(k))→$MACC_{51-48}$ 0→$MACC_{23-0}$
DIML	Load low MAC accumulator with serial input register specified by k.	2J	(SI(k))→$MACC_{23-0}$
DIS	Load data memory with serial input register specified by k.	2B	(SI(k))→dma
DOMH	Load serial output register specified by k with high MAC accumulator	2J	$(MACC)_{47-24}$→SO(k)
DOML	Load serial output register specified by k with low MAC accumulator	2J	$(MACC)_{23-0}$→SO(k)
DOS	Load serial output register specified by k with data memory	2A	(dma)→SO(k)
LPC†	Load HOST registers with coefficient memory <TMS57002 only>	2C	$(cma)_{31-24}$→HOST0 $(cma)_{23-16}$→HOST1 $(cma)_{15-8}$→HOST2 $(cma)_{7-0}$→HOST3
LPD†	Load HOST registers with data memory <TMS57002 only>	2A	$(dma)_{23-16}$→HOST0 $(dma)_{15-8}$→HOST1 $(dma)_{7-0}$→HOST2

† This instruction is specific to the TMS57002 instruction set.

Table 4–7. Instruction Set Summary (Continued)

Mnemonic	Description	Category	Operation
	Input/Output Instructions (Continued)		
RDE†	Load from CMEM the offset address into external RAM to the offset address register. Load read data buffer with external RAM data. <TMS57002 only>	1C	$(cma)_{18-0}$→XOA external RAM data→XRD
REF†	External RAM refresh <TMS57002 only>	2N	(PC)+1→PC
SRBD†	Load data memory with read data buffer. <TMS57002 only>	2B	(XRD)→dma
WRE†	Load from CMEM the offset address into external RAM to the offset address register. Load write data buffer with data memory, and write to external memory. <TMS57002 only>	1E	$(cma)_{18-0}$→XOA (dma)→XWR (XWR)→external memory
	Data Memory Transfer Instructions		
SACC	Store ALU accumulator in coefficient memory	2D	(AACC)→cma
SACD	Store ALU accumulator in data memory	2B	(AACC)→dma
SLMH	Store high MAC accumulator (long word) in data memory	2B	$(MACC)_{47-32}$→dma_{23-8} 0→dma_{7-0}
SLML	Store low MAC accumulator (long word) in data memory	2B	$(MACC)_{31-8}$→dma_{23-0}
SMHC	Store high MAC accumulator in coefficient memory	2D	$(MACC)_{47-16}$→cma_{31-0}
SMHD	Store high MAC accumulator in data memory	2B	$(MACC)_{47-24}$→dma_{23-0}
SMLD	Store low MAC accumulator in data memory	2B	$(MACC)_{23-0}$→dma_{23-0}
STD1	Store TREG in data memory bank 1	2N	(TREG)→dma1
	Control Instructions		
IDLE	Idle until synchronization	3	(PC)+1→PC
NOP1	No primary instruction	1N	(PC)+1→PC
NOP2	No secondary instruction	2N	(PC)+1→PC
RAOV	Reset ALU accumulator overflow bit	2N	0→AOV
RMOM	Reset MAC accumulator overflow mode	2N	0→MOVM
RMOV	Reset MAC accumulator overflow bit	2N	0→MOV
RND	Set MAC output rounding mode	2N	mapping(constant)→RND
SCRM	Set coefficient memory read mode	2N	mapping (constant)→CRM
SFAI	Set ALU input shift mode	2N	mapping(constant)→SFAI
SFAO	Set ALU accumulator output shift mode	2N	mapping(constant)→SFAO
SFMA	Set MAC accumulation shift mode	2N	mapping(constant)→SFMA
SFMO	Set MAC accumulator output shift mode	2N	mapping(constant)→SFMO
SMOM	Set MAC accumulator overflow mode	2N	1→MOVM

† This instruction is specific to the TMS57002 instruction set.

4.4.3 Instruction Bus Usage

Table 4–8 shows how the four data buses (A, B, C and D) are used by each instruction. The instructions are listed in alphabetical order.

The buses used by a given instruction are checked with an 'X'. The data sources and destinations are marked with the name of the bus which is used. A data source is marked with a capital letter, the letter specifying the data bus via which the data is transferred. A data destination is marked with a lower case letter, the letter specifying the data bus via which the data is transferred.

For example, in the case of the ABS instruction, the B bus is used for data transfers. As indicated by the capital 'B', AACC is the data source, and as indicated by the lower case 'b', A–In is the destination location.

Table 4–8. Instruction Bus Usage

Instructions		Buses				Memory		ALU			MAC			Registers						
Mnemonic [Category]		A	B	C	D	CMEM	DMEM	AIN	BIN	AACC	CIN	DIN	MACC	SI	SO	TREG	HOST	XOA	XRD	XWR
ABS	[1I]		X					b		B										
ADD(1)	[1A]		X		X		D	b	d	B										
ADD(2)	[1C]		X	X		C		b	c	B										
ADD(3)	[1A]		X		X		D	b	d				B							
ADD(4)	[1C]		X	X		C		b	c				B							
ADD(5)	[1E]			X	X	C	D	c	d											
ADDS	[1A]	X			X		D,A	a	d											
AMAC	[1E]	X	X	X	X	C	D,A	a	d	B	c	b								
AMPY	[1E]	X	X	X	X	C	D,A	a	d	B	c	b								
AND(1)	[1A]		X		X		D	b	d	B										
AND(2)	[1C]		X	X		C		b	c	B										
AND(5)	[1E]			X	X	C	D	c	d											
B	[3]																			
BGZ	[3]																			
BIOZ	[3]																			
BLZ	[3]																			
BNZ	[3]																			
BV	[3]																			
CMPL	[1I]		X					b		B										
DIMH	[2J]				X								d	D						
DIML	[2J]				X								d	D						
DIS	[2B]				X		d							D						
DOMH	[2J]				X								D		d					
DOML	[2J]				X								D		d					
DOS	[2A]				X		D								d					
IDLE	[3]																			
INCD	[2N]																			

Table 4–8. Instruction Bus Usage (Continued)

Instructions		Buses				Memory		ALU			MAC			Registers						
Mnemonic [Category]		A	B	C	D	CMEM	DMEM	AIN	BIN	AACC	CIN	DIN	MACC	SI	SO	TREG	HOST	XOA	XRD	XWR
LACC	[1C]			X		C				c										
LACD	[1A]				X		D			d										
LBRK	[3]																			
LCAA	[2J]				X					D										
LCAC	[3]																			
LCAK	[3]																			
LDPK	[2N]																			
LD0T	[2A]				X		D									d				
LIRA	[2J]				X					D										
LIRK	[3]																			
LMHC	[1C]			X		C							c							
LMHD	[1A]				X		D						d							
LMLD	[1A]				X		D						d							
LPC†	[2C]			X		C											c			
LPD†	[2A]				X		D										d			
MAC(1)	[1E]			X	X	C	D				c	d								
MAC(2)	[1A]		X		X		D			B	b	d								
MAC(3)	[1C]		X	X		C				B	c	b								
MACS	[1C]		X	X		C				B	c	b								
MACU(1)	[1E]			X	X	C	D				c	d								
MACU(2)	[1A]		X		X		D			B	b	d								
MPY(1)	[1E]			X	X	C	D				c	d								
MPY(2)	[1C]		X	X		C				B	c	b								
MPY(3)	[1A]				X		D					d								
MPYU	[1E]			X	X	C	D				c	d								
NEG	[1I]		X					b		B										
NOP1	[1N]																			
NOP2	[2N]																			
OR(1)	[1A]		X		X		D	b	d	B										
OR(2)	[1C]		X	X		C		b	c	B										
OR(5)	[1E]			X	X	C	D	c	d											
RAOV	[2N]																			
REF†	[2N]																			
RDE†	[1N]			X		C												c		
RMOM	[2N]																			
RMOV	[2N]																			
RND	[2N]																			
RPTK	[3]																			

† This instruction is specific to the TMS57002 instruction set.

Table 4–8. Instruction Bus Usage (Continued)

Instructions	Buses				Memory		ALU			MAC			Registers						
Mnemonic [Category]	A	B	C	D	CMEM	DMEM	AIN	BIN	ACC	CIN	DIN	MACC	SI	SO	TREG	HOST	XOA	XRD	XWR
SACC [2D]			X		c				C										
SACD [2B]				X		d			D										
SCRM [2N]																			
SFAI [2N]																			
SFAO [2N]																			
SFMA [2N]																			
SFML [1N]																			
SFMO [2N]																			
SFMR [1N]																			
SLMH [2B]				X		d						D							
SLML [2B]				X		d						D							
SMHC [2D]			X		c							D							
SMHD [2B]				X		d						C							
SMLD [2B]				X		d						D							
SMOM [2N]																			
SRBD† [2B]	X					d													
STD1 [2N]	X					a												D	
SUB(1) [1A]		X		X		D	b	d	B						A				
SUB(2) [1C]		X	X		C		b	c	B										
SUB(3) [1A]		X		X		D	b	d											
SUB(4) [1C]		X	X		C		b	c				B							
SUB(5) [1E]			X	X	C	D	c	d				B							
WRE† [1E]			X	X	C	D											c		d
XOR(1) [1A]		X		X		D	b	d	B										
XOR(2) [1C]		X	X		C		b	c	B										
XOR(5) [1E]			X	X	C	D	c	d											
ZACC [1N]																			
ZMAC [1N]																			

† This instruction is specific to the TMS57002 instruction set.

4.5 Individual Instruction Descriptions

Each instruction in the instruction set summary is described in the following pages. Instructions are listed in alphabetical order. Information, such as assembler syntax, operands, opcode, execution, descriptions, category, cycles, repeatability and examples, is provided for each instruction. An example instruction is provided on the next two pages to familiarize the user with the instruction description page format.

There are six instructions which are included only in the TMS57002 instruction set. These instructions are marked clearly as being available only for the TMS57002. All other instructions are included in the TMC57000, TMC57001 and TMS57002 instruction sets.

Syntax

Direct: **ADD D(<dma>), A**
Indirect: **ADD {D*/D*+}, A**
Immediate: **SFAI<constant>**

Each program line contains an instruction mnemonic followed by zero, one or two operands. Labels are optional and are place before the instruction mnemonic. An optional comment field may conclude the syntax of the program line. Spaces are required between each field (label, instruction mnemonic, operands and comment field).

Any syntax in boldface is required syntax. Syntax in boldface must be inputted exactly as written or in the case of <>, the information described in the <> must be inputted. A comma is required syntax between operands.

Operands

$0 \leq dma \leq 255$
$0 \leq dma \leq 31$ (DMEM1 of TMS57002 only)
$0 \leq cma \leq 255$
$0 \leq pma \leq 255$
$0 \leq constant \leq 3$

The above set of operands is not comprehensive; however, they are the most frequently used in the instruction set. Operands may be constants or assembly–time expressions referring to memory, I/O ports, shift counts, etc.

Operation

$(PC) + 1 \rightarrow PC$
$(dma)_{23-0} + (AACC)_{31-0} \rightarrow AACC_{31-0}$

Affected by DBP, SFAI, SFAO.
Affects AOV.

An example of the ADD(1) instruction operation sequence is provided above. The instruction operation sequence describes the processing that takes place when the instruction is executed. Any control bits in the 5700x status registers which affect the operation of the instruction are listed. in addition, those control bits affected by the execution of the instruction are also provided.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

The opcode breaks down the various bit fields that make up each instruction word.

Description

Instruction execution and its effect on the rest of the processor or memory contents are described. Any constraints on the operands imposed by the processor or the assembler are discussed. The description parallels and supplements the information provided in other chapters of this user's manual.

Category

1A

Please refer to Section 4.1 a description of categories.

Cycles

1 (Class 2)

This field specifies the number of cycles required to execute the instruction. The cycle timings are for single–instruction execution, not for repeat mode. Refer to Table 3–3, Section 3.6.2 for a summary of classes.

Repeatable

yes

This field specifies whether an instruction is repeatable or not.

Example

Example code is included for each instruction. The effect of the code on memory and/or registers is summarized.

Syntax **ABS**

Operands None

Operation
(PC) + 1 → PC
|(AACC)| → AACC

Affected by SFAO.
Affects AOV.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Description

The ALU accumulator (AACC) output is shifted as specified by SFAO; if this shifted value is less than zero, the contents of the AACC are replaced by the twos–complement of this shifted value. Otherwise, the value of the AACC is unchanged by the execution of this instruction.

Note that if after shifting the AACC output as specified by SFAO, the result is 80000000h, this is a special case. The ABS of 80000000h is 80000000h and AOV (AACC overflow flag) is set to one. AOV is cleared to zero at synchronization, reset or with the BV, INCD or RAOV instructions. For a description of how the SFAO control bit affects the operand, refer to the SFAO instruction description.

Category 1I

Cycles 1 (Class 2)

Repeatable no

Example 1

```
ABS ;SFAO=0
```

Before Instruction:			After Instruction:		
AACC	00001234h	AOV x	AACC	00001234h	AOV x

Example 2

```
ABS ;SFAO=0
```

Before Instruction:			After Instruction:		
AACC	FFFFFFFFh	AOV x	AACC	00000001h	AOV x

Example 3

```
ABS ;SFAO=0
```

Before Instruction:			After Instruction:		
AACC	80000000h	AOV x	AACC	80000000h	AOV 1

Syntax
Direct: **ADD D(<dma>), A**
Indirect: **ADD {D*/D*+}, A**

Operands
0 ≤ dma ≤ 255
0 ≤ dma ≤ 31 (DMEM1 of TMS57002 only)

Operation
(PC) + 1 → PC
$(dma)_{23-0} + (AACC)_{31-0} \rightarrow AACC_{31-0}$

Affected by DBP, SFAI, SFAO.
Affects AOV.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed data memory location are right–shifted as specified by SFAI, the output of the ALU accumulator (AACC) is shifted as specified by SFAO; these shifted values are added and stored in the AACC.

In the case of overflow, AOV (AACC overflow flag) is set to one. AOV is cleared to zero at synchronization, reset or with the BV, INCD or RAOV instructions.

For a description of how the DBP, SFAI and SFAO control bits affect the operands, refer to the LDPK, SFAI and SFAO instruction descriptions, respectively.

Category 1A

Cycles 1 (Class 2)

Repeatable yes

Example

Given Conditions:BA0=0, DBP=0, SFAI=0, SFAO=0.

```
ADD D(86), A
or
ADD D*, A ;ID=86.
```

Case 1:

Before Instruction:		After Instruction:	
DMEM0 86	000003h	DMEM0 86	000003h
AACC	00000156h	AACC	00000456h
AOV	X	AOV	X

Case 2:

Before Instruction:		After Instruction:	
DMEM0 86	000003h	DMEM0 86	000003h
AACC	7FFFFD00h	AACC	80000000h
AOV	X	AOV	1

Syntax

Direct: **ADD C(<cma>), A**
Indirect: **ADD {C*/C*+}, A**

Operands

$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(cma)_{31-0} + (AACC)_{31-0} \rightarrow AACC_{31-0}$

Affected by CRM, SFAO.
Affects AOV.

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11	10 9 8 7 6 5 4 3 2 1 0
0 0 0 1 0 0	X X X X X X X	Memory Address. See Sections 4.2/4.3

Description

The data of the addressed coefficient memory location are provided by CRM, the output of the ALU accumulator (AACC) is shifted as specified by SFAO; these two values are added and stored in the AACC.

In the case of overflow, AOV (AACC overflow flag) is set to one. AOV is cleared to zero at synchronization, reset or with the BV, INCD or RAOV instructions.

For a description of how the CRM and SFAO control bits affect the operands, refer to the SCRM and SFAO instruction descriptions, respectively.

Category

1C

Cycles

1 (Class 2)

Repeatable

yes

Example

Given Conditions:CRM=0, SFAO=0.

```
ADD C(125), A
or
ADD C*, A ;CA=125.
```

Case 1:

	Before Instruction:		After Instruction:	
CMEM 125	00000003h		00000003h	
AACC	00000156h	AOV x	00000159h	AOV x

Case 2:

	Before Instruction:		After Instruction:	
CMEM 125	00000003h		00000003h	
AACC	7FFFFFFDh	AOV x	80000000h	AOV 1

Syntax

Direct: **ADD D(<dma>), M**
Indirect: **ADD {D*/D*+}, M**

Operands

$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(dma)_{23-0} + (MACC)_{47-16} \rightarrow AACC_{31-0}$

Affected by DBP, MOVM, RND, SFAI, SFMO.
Affects AOV, MOV .

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 0 0 1 0 1	X X X X X X X Memory Address. See Sections 4.2/4.3

Description

The data of the addressed data memory location are right–shifted as specified by SFAI, the output of the MAC accumulator (MACC) is shifted and rounded by SFMO and RND respectively; these two shifted values are added and the result is stored in the ALU accumulator (AACC).

In the case of overflow, AOV (AACC overflow flag) is set to one. AOV is cleared to zero at synchronization, reset or with the BV, INCD or RAOV instructions.

In the case of overflow when shifting or rounding the MACC output value, MOV (MACC overflow flag) is set to one. If MOVM=1, the MACC operand is clipped to 7FFFFFFFFFFFh or 800000000000h and then rounded as specified by RND. Refer to the SMOM instruction description for further details. If MOVM=0, the MACC output value is not clipped. MOV is cleared to zero at synchronization, reset or with the RMOV or INCD instructions.

For a description of how the DBP, MOVM, SFAI, SFMO and RND control bits affect the operands, refer to the LDPK, SMOM, SFAI, SFMO and RND instruction descriptions, respectively.

Category 1A

Cycles 1 (Class 2)

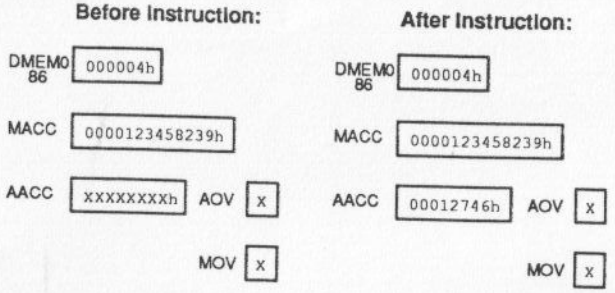
Repeatable yes

Example

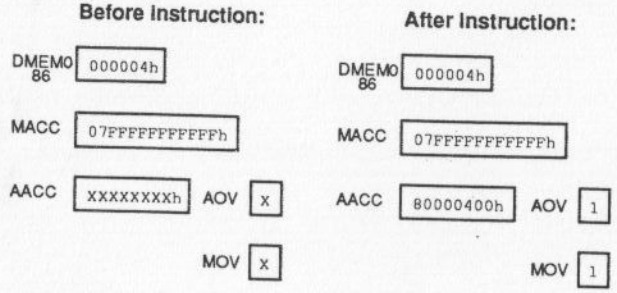
Given Conditions:BA0=0, DBP=0, RND=1 SFAI=0, SFMO=0.

```
ADD D(86), M
or
ADD D*, M ;ID=86.
```

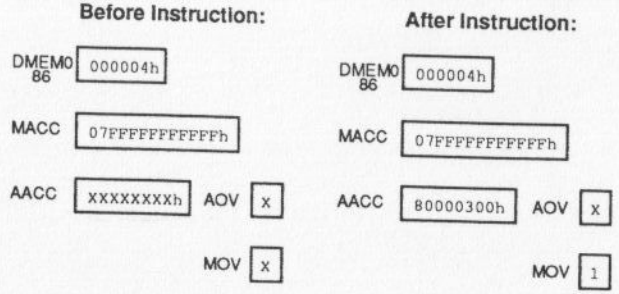
Case 1:



Case 2: (MOVM=0)



Case 3: (MOVM=1)



Syntax

Direct: **ADD C(<cma>), M**
Indirect: **ADD {C*/C*+}, M**

Operands

$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(cma)_{31-0} + (MACC)_{47-16} \rightarrow AACC_{31-0}$

Affected by CRM, MOVM, RND, SFMO.
Affects AOV, MOV .

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 0 0 1 1 0	X X X X X X X X X X X X X X X X X X X X X X	Memory Address. See Sections 4.2/4.3

Description

The data of the addressed coefficient memory location are provided by CRM, the output of the MAC accumulator (MACC) is shifted and rounded as specified by SFMO and RND, respectively; these two values are added and stored in the ALU accumulator.

In the case of overflow, AOV (AACC overflow flag) is set to one. AOV is cleared to zero at synchronization, reset or with the BV, INCD or RAOV instructions.

In the case of overflow when shifting or rounding the MACC output value, MOV (MACC overflow flag) is set to one. If MOVM=1, the MACC operand is clipped to 7FFFFFFFFFFFh or 800000000000h and rounded as specified by RND. Refer to the SFMO instruction description for further details. If MOVM=0, the MACC output value is not clipped. MOV is cleared to zero at synchronization, reset or with the RMOV or INCD instructions.

For a description of how the CRM, MOVM, SFMO and RND control bits affect the operands, refer to the CRM, SMOM, SFMO and RND instruction descriptions respectively.

Category

1C

Cycles

1 (Class 2)

Repeatable

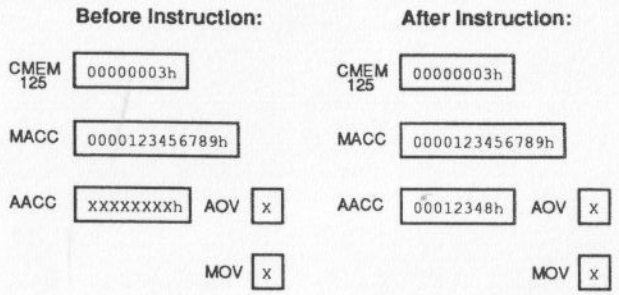
yes

Example

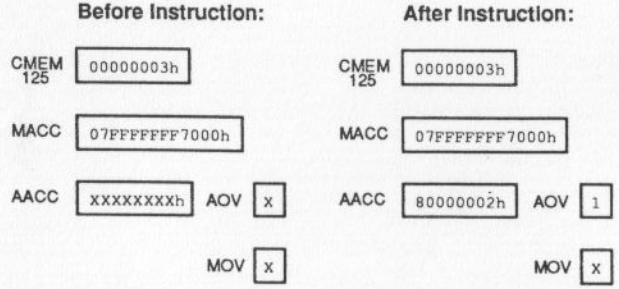
Given Conditions:CRM=0, MOVM=0, RND=1, SFMO=0.

```
ADD C(125), M
or
ADD C*, M ;CA=125.
```

Case 1:



Case 2:



Syntax

Direct: **ADD D(<dma>), {C*/C*+}**
ADD {D*/D*+}, C(<cma>)
Indirect: **ADD {D*/D*+}, {C*/C*+}**

Operands

$0 \le cma \le 255$
$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(dma)_{23-0} + (cma)_{31-0} \rightarrow AACC_{31-0}$

Affected by CRM, DBP, SFAI.
Affects AOV.

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 0 0 1 1 1	X X X X X X X Memory Address. See Sections 4.2/4.3

Description

The data of the addressed data memory location are right–shifted as specified by SFAI, the data of the addressed coefficient memory location are provided by CRM; these two values are added and stored in the ALU accumulator (AACC).

In the case of overflow, AOV (AACC overflow flag) is set to one. AOV is cleared to zero at synchronization, reset or with the BV, INCD or RAOV instructions.

For a description of how the CRM, DBP and SFAI control bits affect the operands, refer to the SCRM, LDPK and SFAI instruction descriptions, respectively.

Category 1E

Cycles 1 (Class 2)

Repeatable yes

Example

Given Conditions:BA0=0, CRM=0, DBP=0, SFAI=0.

```
ADD D(86), C* ;CA=125.
```
or
```
ADD D*, C(125) ;ID=86.
```
or
```
ADD D*, C* ;CA=125.
```

Case 1:

	Before Instruction:		After Instruction:	
DMEM0 86	000004h		DMEM0 86	000004h
CMEM 125	00000123h		CMEM 125	00000123h
AACC	XXXXXXXXh	AOV X	AACC	00000523h AOV X

Case 2:

	Before Instruction:		After Instruction:	
DMEM0 86	000004h		DMEM0 86	000004h
CMEM	7FFFFC00h		CMEM	7FFFFC00h
AACC	XXXXXXXXh	AOV X	AACC	80000000h AOV 1

Syntax Indirect: **ADDS {D*/D*+}**

Operands $0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation $(PC) + 1 \rightarrow PC$
$(dma0) \times 2^{-1} + (dma1) \times 2^{-1} \rightarrow AACC_{31-7}$
$0 \rightarrow AACC_{6-0}$

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11	10 9 8 7 6 5 4 3 2 1 0
0 0 1 0 0 0	X X X X X X X	Memory Address. See Sections 4.2/4.3

Description

The data of the indirectly addressed data memory banks, DMEM0 and DMEM1, are added after a 1 bit right arithmetic shift of both operands at the ALU's Input Shifters. The result is stored in the ALU accumulator (AACC).

Note that the SFAI, the ALU Input Shifters' control bit is ignored during the ADDS instruction. By default, SFAI=1, which shifts the operands 1 bit right at the ALU Input Shifters. Refer to the SFAI instruction description for more details.

In addition, note that the value of the DBP (data bank pointer) control bit is not valid with this instruction, since both data memory banks (DMEM0 and DMEM1) are accessed.

This instruction, in addition to the AMAC, AMPY, LD0T, MACS and STD1 instructions are very useful in implementing linear phase FIR filter calculations. Please refer to Section 5.4, Linear Phase FIR Filters, for more details.

Category 1A

Cycles 1 (Class 2)

Repeatable yes

Example

Given Conditions:BA0=255, BA1=1, ID=86.

```
ADDS D*
```

	Before Instruction:		After Instruction:
DMEM0 85	001234h	DMEM0 85	001234h
DMEM1 87	006789h	DMEM1 87	006789h
AACC	XXXXXXXXh	AACC	003CDE80h

Syntax

Direct: **AMAC {D*/D*+}, C(<cma>), A**
Indirect: **AMAC {D*/D*+}, {C*/C*+}, A**

Operands

$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(cma)_{31-0} \times (AACC)_{31-0} \times 2 + (MACC)_{51-0} \rightarrow MACC_{51-0}$
$(dma0) \times 2^{-1} + (dma1) \times 2^{-1} \rightarrow AACC_{31-7}$
$0 \rightarrow AACC_{6-0}$

Affected by CRM, SFAO and SFMA.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	1	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the specified coefficient memory location are provided by CRM, the higher 25 bits of the AACC output after having been shifted by SFAO are arithmetically shifted 1 bit left by SFMI, the MACC output is shifted as specified by SFMA; the first two values are multiplied and added to the third. In parallel, the data of the indirectly addressed data memory banks, DMEM0 and DMEM1, are added after a 1 bit right arithmetic shift of both operands at the ALU's Input Shifters.

Note that the SFAI, the ALU Input Shifters' control bit is ignored during the AMAC instruction. By default, SFAI=1, which shifts the operands 1 bit right at the ALU Input Shifters. Refer to the SFAI instruction description for more details.

In addition, note that the value of the DBP (data bank pointer) control bit is not valid with this instruction, since both data memory banks (DMEM0 and DMEM1) are accessed.

This instruction, in addition to the ADDS, AMPY, LD0T, MAC and STD1 instructions are very useful in implementing linear phase FIR filter calculations.

For a description of how the CRM, SFAO and SFMA control bits affect the operands, refer to the SCRM, SFAO and SFMA instruction descriptions, respectively.

Note:

Take care in choosing the instruction that follows an AMAC instruction. The AMAC execution is pipelined. This instruction requires one machine cycle but the result is not valid in time for the following instruction to use the result for calculation unless the AMAC instruction is followed by a MAC, MACS, MACU, AMAC, AMPY, SFML or SFMR instruction. For all other instructions, the result is valid in time for the second instruction from the AMAC instruction to use the result for calculation.

Category

1E

Cycles

2 (Class 4)

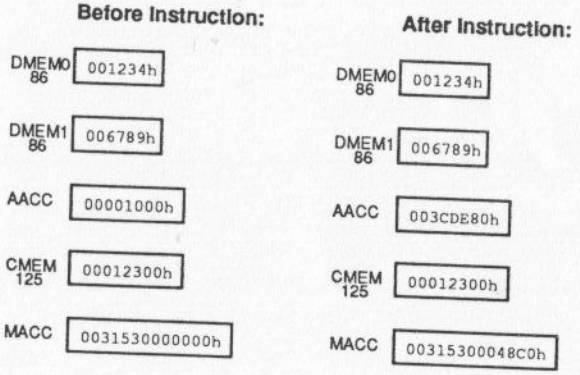
Repeatable

yes

Example

Given Conditions:CRM=0, BA0=0, BA1=0, SFAO=0, SFMA=0.

```
AMAC D*+, C(125), A ;ID=86.
or
AMAC D*+, C*+, A ;CA=125, ID=86.
```



Syntax

Direct: **AMPY {D*/D*+}, C(<cma>), A**
Indirect: **AMPY {D*/D*+}, {C*/C*+}, A**

Operands

$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(cma)_{31-0} \times (AACC)_{31-0} \times 2^{-1} \rightarrow MACC_{31-7}$
$(dma0) \times 2^{-1} + (dma1) \times 2^{-1} \rightarrow AACC_{31-7}$
$0 \rightarrow AACC_{6-0}$

Affected by CRM and SFAO.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	0	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the specified coefficient memory location are provided by CRM, the higher 25 bits of the AACC output after having been shifted by SFAO are arithmetically shifted 1 bit left by SFMI; the two values are multiplied together. In parallel, the data of the indirectly addressed data memory banks, DMEM0 and DMEM1, are added after a 1 bit right arithmetic shift of both operands at the ALU's Input Shifters.

Note that the SFAI, the ALU Input Shifters' control bit, is ignored during the AMPY instruction. By default, SFAI=1, which shifts the operands 1–bit right at the ALU Input Shifters. Refer to the SFAI instruction description for more details.

In addition, note that the value of the DBP (data bank pointer) control bit is not valid with this instruction, since both data memory banks (DMEM0 and DMEM1) are accessed.

This instruction, in addition to the ADDS, AMAC, LD0T, MACS and STD1 instructions are very useful in implementing linear phase FIR filter calculations.

For a description of how the CRM and SFAO control bits affect the operands, refer to the SCRM and SFAO instruction descriptions, respectively.

Note:

Take care in choosing the instruction that follows an AMPY instruction. The AMPY execution is pipelined. This instruction requires one machine cycle but the result is not valid in time for the following instruction to use the result for calculation unless the AMPY instruction is followed by a MAC, MACS, MACU, AMAC, AMPY, SFML or SFMR instruction. For all other instructions, the result is valid in time for the second instruction from the AMPY instruction to use the result for calculation.

Cycles

2 (Class 4)

Category

1E

Repeatable

yes

Example

Given Conditions: BA0=255, BA1=1, CRM=0, SFAO=0, FMA=0.

```
AMPY D*+, C(125), A ;ID=86.
or
AMPY D*+, C*+, A ;CA=125, ID=86.
```

	Before Instruction:	After Instruction:
DMEM0 85	001234h	001234h
DMEM1 87	006789h	006789h
AACC	00001000h	003CDE80h
CMEM 125	00012300h	00012300h
MACC	0031530000000h	00000000048C0h

Syntax

Direct: **AND D(<dma>), A**
Indirect: **AND {D*/D*+}, A**

Operands

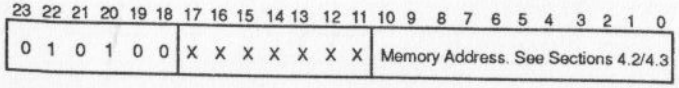
$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(dma)_{23-0}$ AND $(AACC)_{31-8} \rightarrow AACC_{31-8}$
$0 \rightarrow AACC_{7-0}$

Affected by DBP, SFAI, SFAO.

Encoding



Description

The data of the addressed data memory location are right–shifted as specified by SFAI, the output of the ALU accumulator (AACC) is shifted as specified by SFAO; these two values are ANDed bitwise and the result is stored in the AACC. Note that the lower 8 bits of the shifted AACC output value are ANDed bitwise with all zeroes.

For a description of how the DBP, SFAI and SFAO control bits affect the operands, refer to the LDPK, SFAI and SFAO instruction descriptions, respectively.

Category

1A

Cycles

1 (Class 2)

Repeatable

no

Example

Given Conditions:BA0=0, DBP=0, SFAI=0, SFAO=0.

```
AND D(51), A
or
AND D*, A ;ID=51.
```

Before Instruction:

DMEM0 51: 00FFFFh
AACC: 12345678h

After Instruction:

DMEM0 51: 00FFFFh
AACC: 00345600h

Syntax

Direct: **AND C(<cma>), A**
Indirect: **AND {C*/C*+}, A**

Operands

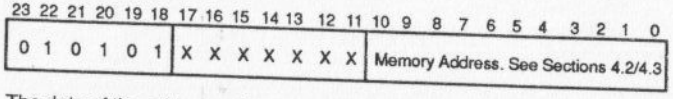
$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(cma)_{31-0}$ AND $(AACC)_{31-0} \rightarrow AACC_{31-0}$

Affected by CRM, SFAO.

Encoding



Description

The data of the addressed coefficient memory location are provided by CRM, the output of the ALU accumulator (AACC) is shifted as specified by SFAO; these two values are ANDed bitwise and the result is stored in the AACC.

For a description of how the CRM and SFAO control bits affect the operands, refer to the SCRM and SFAO instruction descriptions, respectively.

Category

1C

Cycles

1 (Class 2)

Repeatable

no

Example

Given Conditions:CRM=0, SFAO=0.

```
AND C(187), A
or
AND C*, A ;CA=187.
```

Before Instruction:

CMEM 187: 0000FFFFh
AACC: 12345678h

After Instruction:

CMEM 187: 0000FFFFh
AACC: 00005678h

Syntax

Direct: **AND D(<dma>), {C*/C*+}**
AND {D*/D*+}, C(<cma>)
Indirect: **AND {D*/D*+}, {C*/C*+}**

Operands

$0 \le cma \le 255$, $0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(dma)_{23-0}$ AND $(cma)_{31-8} \rightarrow AACC_{31-8}$
$0 \rightarrow AACC_{7-0}$

Affected by CRM, DBP, SFAI.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	0	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed data memory location are right–shifted as specified by SFAI, the data of the addressed coefficient memory location are provided by CRM; these two values are ANDed bitwise and the result is stored in the ALU accumulator (AACC). Note that the lower 8 bits of the coefficient memory value are ANDed bitwise with all zeroes.

For a description of how the SFAI and CRM control bits affect the operands, refer to the SFAI and SCRM instruction descriptions, respectively.

Category

1E

Cycles

1 (Class 2)

Repeatable

no

Example

Given Conditions:BA0=0, CRM=0, DBP=0, SFAI=0.

```
AND D(86), C* ;CA=125.
```
or
```
AND D*, C(125) ;ID=86.
```
or
```
AND D*, C* ;CA=125.
```

Before Instruction:		After Instruction:	
DMEM0 86	000FFFh	DMEM0 86	000FFFh
CMEM 125	12345678h	CMEM 125	12345678h
AACC	XXXXXXXXh	AACC	00045600h

Syntax

B <pma>

Operands

$0 \le pma \le 255$

Operation

$(pma) \rightarrow PC$

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	0	0	1	0	0	0	X	X	X	Program Memory Address							

Description

Control passes to the designated program memory address (pma). Pma can be either a symbolic or numeric address.

Category

3

Cycles

3 (Class 5)

Repeatable

no

Example

Given Conditions:The label PRG126 is equal to 126.

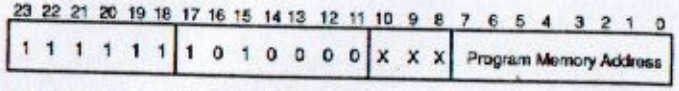
```
B PRG126    ;The program counter is set to 126 and
            ;control passes to line 126.
```

Syntax **BGZ <pma>**

Operands 0 ≤ pma ≤ 255

Operation
If (AACC) > 0:
Then (pma) → PC
Else (PC) + 1 → PC

Encoding



Description If the ALU accumulator (AACC) is greater than zero, control passes to the designated program memory address (pma). Otherwise, the program counter is incremented by one and control passes to the next instruction. Pma can be either a symbolic or numeric address.

Category 3

Cycles 3 if branch, 1 if no branch (Class 5)

Repeatable no

Example Given Conditions:The label `PRG165` is equal to 165.

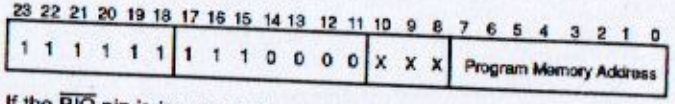
```
BGZ PRG165  ;If the AACC is greater than zero, the
            ;program counter is set to 165 and control
            ;passes to line 165. If the  AACC is less
            ;than or equal to zero, the program counter
            ;is incremented by one and control passes
            ;to the next instruction.
```


Syntax **BIOZ <pma>**

Operands 0 ≤ pma ≤ 255

Operation
If $\overline{BIO}$ = 0:
Then (pma) → PC
Else (PC) + 1 → PC

Encoding



Description If the $\overline{BIO}$ pin is low, control passes to the designated program memory address (pma). Otherwise, the program counter is incremented by one and control passes to the next instruction. Pma can be either a symbolic or numeric address.

Category 3

Cycles 3 if branch, 1 if no branch (Class 5)

Repeatable no

Example Given Conditions:The label `PRG229` is equal to 229.

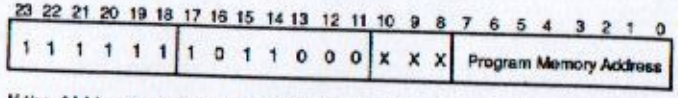
```
BIOZ PRG229 ;If the BIO pin is equal to zero, the
            ;program counter is set to 229 and control
            ;passes to line 229. If the BIO pin is
            ;equal to one, the program counter is
            ;incremented by one and control passes to
            ;the next instruction.
```

Syntax **BLZ <pma>**

Operands 0 ≤ pma ≤ 255

Operation
If (AACC) < 0:
Then (pma) → PC
Else (PC) + 1 → PC

Encoding



Description If the ALU accumulator (AACC) is less than zero, control passes to the designated program memory address (pma). Otherwise, the program counter is incremented by one and control passes to the next instruction. Pma can be either a symbolic or numeric address.

Category 3

Cycles 3 if branch, 1 if no branch (Class 5)

Repeatable no

Example Given Conditions:The label PRG92 is equal to 92.

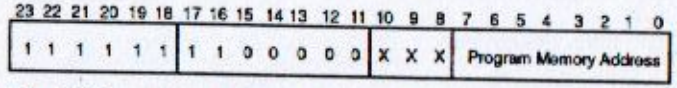
```
BLZ PRG92   ;If the AACC is less than zero,the program
            ;counter is set to 92 and control passes to
            ;line 92. If the AACC is greater than or
            ;equal to zero, the program counter is
            ;incremented by one and control passes to
            ;the next instruction.
```


Syntax **BNZ <pma>**

Operands 0 ≤ pma ≤ 255

Operation
If (AACC) ≠ 0:
Then (pma) → PC
Else (PC) + 1 → PC

Encoding



Description If the ALU accumulator (AACC) is not equal to zero, control passes to the designated program memory address (pma). Otherwise, the program counter is incremented by one and control passes to the next instruction. Pma can be either a symbolic or numeric address.

Category 3

Cycles 3 if branch, 1 if no branch (Class 5)

Repeatable no

Example Given Conditions:The label PRG72 is equal to 72.

```
BNZ PRG72   ;If the AACC is not equal to zero,the
            ;program counter is set to 72 and control
            ;passes to line 72. If the AACC is equal to
            ;zero, the program counter is incremented
            ;by one, and control passes to the next
            ;instruction.
```

Syntax **BV <pma>**

Operands $0 \le pma \le 255$

Operation
If (AOV) = 1:
Then (pma) → PC
Else (PC) + 1 → PC

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	0	0	0	X	X	X	Program Memory Address							

Description If AOV (AACC overflow flag) is equal to one, control passes to the designated program memory address (pma) and AOV is cleared to zero. Otherwise, the program counter is incremented by one and control passes to the next instruction. Pma can be either a symbolic or numeric address.

Category 3

Cycles 3 if branch, 1 if no branch (Class 5)

Repeatable no

Example Given Conditions:The label `PRG215` is equal to 215.

```
BV PRG215   ;If AOV is equal to one, the program
            ;counter is set to 215 and control passes
            ;to line 215. If AOV is equal to zero, the
            ;program counter is incremented by one and
            ;control passes to the next instruction.
```


Syntax **CMPL**

Operands None

Operation
$\overline{(PC) + 1} \rightarrow PC$
$\overline{(AACC)}_{31-0} \rightarrow AACC_{31-0}$

Affected by SFAO.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Description The output of the ALU accumulator (AACC) is shifted as specified by SFAO; the contents of the AACC are replaced with the logical inversion (ones–complement) of the shifted AACC output value.

For a description of how the SFAO control bit affects the operand, refer to the SFAO instruction description.

Category 1I

Cycles 1 (Class 2)

Repeatable no

Example 1

```
CMPL ;SFAO=0
```

	Before Instruction:	After Instruction:
AACC	F7982513h	0867DAECh

Example 2

```
CMPL ;SFAO=0
```

	Before Instruction:	After Instruction:
AACC	0001234h	FFFEDCBh

Syntax **DIMH <constant>**

Operands $0 \le \text{constant} \le 3$

Operation

$(PC) + 1 \rightarrow PC$
$SI(constant) \rightarrow MACC_{47-24}$
$(SI(constant))_{23} \rightarrow MACC_{51-48}$
$0 \rightarrow MACC_{23-0}$

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	0	1	0	1	K	K	X	X	X	X	X	X	X	X	X	X	X

Description

The data of the specified SI register (serial input port register) are loaded into bits 47 through 24 of the MAC accumulator (MACC). The 4–bit head margin of the MACC, bits 51 through 48, is sign extended and bits 23 through 0 are filled with zeroes. The SI register used as an operand is specified by the value of 'constant' as shown below:

Constant	Register
0	SI0_L
1	SI0_R
2	SI1_L
3	SI1_R

When using both the DIMH and DIML instructions to load a 48–bit value to the MACC, be certain to execute DIMH first since this instruction clears the lower bits to zero.

In order to avoid reading the previously sampled data or the following data sample, instruction execution timing and serial input timing must be coordinated correctly. Refer to Section 3.8.1, Serial Input Port, for more details.

Category 2J

Cycles 1 (Class 1)

Repeatable no

Example

```
DIMH 0
```

	Before Instruction:	After Instruction:
SI0_L	F4B123h	F4B123h
MACC	XXXXXXXXXXXXh	FF4B123000000h

Syntax **DIML <constant>**

Operands $0 \le \text{constant} \le 3$

Operation

$(PC) + 1 \rightarrow PC$
$SI(constant) \rightarrow MACC_{23-0}$
$MACC_{51-24} \rightarrow MACC_{51-24}$

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	0	1	1	0	K	K	X	X	X	X	X	X	X	X	X	X	X

Description

The contents of the SI register (serial input port register) are loaded into bits 23 through 0 of the MAC accumulator (MACC). The higher bits of the MACC remain unchanged.The SI register used as an operand is specified by the value of 'constant' as shown below:

Constant	Register
0	SI0_L
1	SI0_R
2	SI1_L
3	SI1_R

When using both the DIMH and DIML instructions to load a 48–bit value to the MACC, be certain to execute DIMH first since this instruction clears the lower bits to zero.

In order to avoid reading the previously sampled data or the following data sample, instruction execution timing and serial input timing must be coordinated correctly. Refer to Section 3.8.1, Serial Input Port, for more details.

Category 2J

Cycles 1 (Class 1)

Repeatable no

Example

```
DIML 0
```

	Before Instruction:	After Instruction:
SI0_L	456789h	456789h
MACC	FF4B123000000h	FF4B123456789h

Syntax

Direct: **DIS <constant>, D(<dma>)**
Indirect: **DIS <constant>, {D*/D*+}**

Operands

0 ≤ dma ≤ 255
0 ≤ dma ≤ 31 (DMEM1 of TMS57002 only)
0 ≤ constant ≤ 3

Operation

(PC) + 1 → PC
SI(constant) → dma_{23-0}

Affected by DBP.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	0	1	0	0	K	K	X	X	X	X	X	X	X	X	X	X	X

Description

The contents of the specified SI register (serial input port register) are loaded into the addressed data memory location. The SI register used as an operand is specified by the value of 'constant' as shown below:

Constant	Register
0	SI0_L
1	SI0_R
2	SI1_L
3	SI1_R

In order to avoid reading the previously sampled data or the following data sample, instruction execution timing and serial input timing must be coordinated correctly. Refer to Section 3.8.1, Serial Input Port, for more details.

Category

2B

Cycles

1 (Class 1)

Repeatable

no

Example

Given Conditions:BA0=0, DBP=0.

```
DIS 2, D(39)
or
DIS 2, D* ;ID=39.
```

Before Instruction:		After Instruction:	
SI1_L	F4B100h	SI1_L	F4B100h
$DMEM0_{39}$	XXXXXh	$DMEM0_{39}$	F4B100h

Syntax

DOMH <constant>

Operands

0 ≤ constant ≤ 3

Operation

(PC) + 1 → PC
$(MACC)_{47-24}$ → SO(constant)

Affected by MOVM, RND and SFMO.
Affects MOV.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	1	0	0	0	K	K	X	X	X	X	X	X	X	X	X	X	X

Description

The output of the MAC accumulator (MACC) is shifted and rounded as specified by SFMO and RND, respectively; the higher 24 bits of this value are loaded into the specified SO register (serial output register). Refer to the SFMO and RND instruction descriptions for more details on how these control bits affect the operand.The SO register used as an operand is specified by the value of 'constant' as shown below:

Constant	Register
0	SO0_L
1	SO0_R
2	SO1_L
3	SO1_R

If overflow is detected (MOV=1) and MOVM=1, depending upon the direction of overflow, the value is clipped to 7FFFFFFFFFFFh or 800000000000h, rounded as specified by RND and bits 47–24 are loaded into the specified SO register instead of the MACC output value. If MOVM=0, the MACC output value is not clipped.

In order to avoid reading the previously sampled data or the following data sample, instruction execution timing and serial output port timing must be coordinated correctly. Refer to Section 3.8.2, Serial Output Port, for more details.

Category

2J

Cycles

1 (Class 1)

Repeatable

no

Example

Given Conditions:MOVM=1, RND=0, SFMO=0.

```
DOMH  0
```

	Before instruction:	After instruction:
MACC	FF4B123456789h	FF4B123456789h
SO0_L	XXXXXXh	F4B123h

Syntax **DOML <constant>**

Operands $0 \le \text{constant} \le 3$

Operation

$(PC) + 1 \rightarrow PC$

$(MACC)_{23-0} \rightarrow SO(\text{constant})$

Affected by MOVM.
Affects MOV.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	1	0	0	1	K	K	X	X	X	X	X	X	X	X	X	X	X

Description

Bits 23 through 0 of the MAC accumulator (MACC) are loaded into the specified SO register (serial output port register). The SO register used as an operand is specified by the value of 'constant' as shown below:

Constant	Register
0	SO0_L
1	SO0_R
2	SO1_L
3	SO1_R

Although the MACC output is not affected by the Output Shifter and the Output Rounder, overflow detection is performed. The number of MACC bits checked for overflow is defined by the value of SFMO. Refer to the SFMO instruction for further details.

In the case that overflow is detected (MOV=1) and MOVM=1, depending upon the direction of overflow, the value is clipped to either 7FFFFFFFFFFh or 800000000000h and bits 23–0 are loaded into the specified SO register instead of the MACC output value. If MOVM=0, the MACC output value is not clipped. Note that with this instruction, there is no rounding of a clipped result. See the SMOM instruction description for more details.

In order to avoid reading the previously sampled data or the following data sample, instruction execution timing and serial output timing must be coordinated correctly. Refer to Section 3.8.2, Serial Output Port, for more details.

Category 2J

Cycles 1 (Class 1)

Repeatable no

Example Given Conditions:MOVM=1, RND=0, SFMO=0.

```
DOML 3
```

	Before Instruction:	After Instruction:
MACC	FF4B123456789h	FF4B123456789h
SO1_R	XXXXXXh	456789h

Syntax Direct: **DOS <constant>, D(<dma>)**
Indirect: **DOS <constant>, {D*/D*+}**

Operands
$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)
$0 \le constant \le 3$

Operation
$(PC) + 1 \rightarrow PC$
$(dma)_{23-0} \rightarrow SO(constant)$

Affected by DBP.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	0	1	1	1	K	K	X	X	X	X	X	X	X	X	X	X	X

Description The data of the addressed data memory location are loaded into the specified SO register (serial output port register). The SO register used as an operand is specified by the value of 'constant' as shown below:

Constant	Register
0	SO0_L
1	SO0_R
2	SO1_L
3	SO1_R

In order to avoid reading the previously sampled data or the following data sample, instruction execution timing and serial output timing must be coordinated correctly. Refer to Section 3.8.2, Serial Output Port, for more details.

Category 2A

Cycles 1 (Class 1)

Repeatable no

Example Given Conditions:BA0=0, DBP=0.

```
DOS 1, D(40)
or
DOS 1, D* ;ID=40.
```

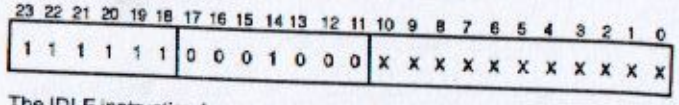
	Before Instruction:	After Instruction:
DMEM0 40	123400h	123400h
SO0_R	XXXXXXh	123400h

Syntax **IDLE**

Operands None

Operation (PC) + 1 → PC
Halt program execution until synchronization or reset.

Encoding



Description The IDLE instruction forces the program being executed to wait until synchronization or reset occurs. The program counter (PC) is incremented only once, and the device then remains in an idle state.

Category 3

Cycles 1 (Class 8)

Repeatable no

Example

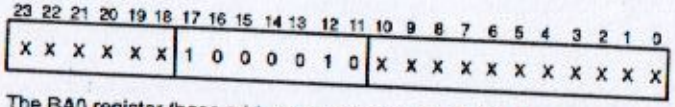
```
IDLE  ;The processor idles until reset or
      ;synchronization.
```


Syntax **INCD**

Operands None

Operation (PC) + 1 → PC
(BA0)−1 → BA0
(BA1)+1 → BA1

Encoding



Description The BA0 register (base address register for DMEM0) is decremented by one and the BA1 register (base address register for DMEM1) is incremented by one. The INCD instruction resets AOV (AACC overflow flag) and MOV (MACC overflow flag) to zero.

Synchronization has the same effect on the BA0 and BA1 registers as the INCD instruction.

Category 2N

Cycles 1 (Class 1)

Repeatable yes

Example

```
INCD
```

	Before Instruction:	After Instruction:
BA0	12h	11h
BA1	56h	57h
AOV	1	0
MOV	1	0

Syntax

Direct: **LACC C(<cma>)**
Indirect: **LACC {C*/C*+}**

Operands

$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(cma)_{31-0} \rightarrow AACC_{31-0}$

Affected by CRM.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed coefficient memory location are provided by CRM and loaded to the ALU accumulator (AACC).

For a description of how the CRM control bit affects the operand, refer to the SCRM instruction description.

Category

1C

Cycles

1 (Class 2)

Repeatable

no

Example

Given Conditions:CRM=0.

```
LACC C(125)
or
LACC C* ;CA=125.
```

	Before Instruction:	After Instruction:
CMEM 125	00000123h	00000123h
AACC	XXXXXXXXh	00000123h

Syntax

Direct: **LACD D(<dma>)**
Indirect: **LACD {D*/D*+}**

Operands

$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(dma)_{23-0} \rightarrow AACC_{31-8}$
$0 \rightarrow AACC_{7-0}$

Affected by DBP, SFAI.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed data memory location are right–shifted as specified by SFAI and loaded to the higher 24 bits of the ALU accumulator (AACC). The lower 8 bits of the AACC are loaded with zeroes.

For a description of how the DBP and SFAI control bits affect the operands, refer to the LDPK and SFAI instruction descriptions.

Category

1A

Cycles

1 (Class 2)

Repeatable

no

Example

Given Conditions:BA0=0, DBP=0.

```
LACD D(86)
or
LACD D* ;ID=86.
```

Case 1:(SFAI=0)

Before Instruction:		After Instruction:	
DMEM0 86	FFFFFFh	DMEM0 86	FFFFFFh
AACC	XXXXXXXXh	AACC	FFFFFF00h

Case 2:(SFAI=1)

Before Instruction:		After Instruction:	
DMEM0 86	FFFFFFh	DMEM0 86	FFFFFFh
AACC	XXXXXXXXh	AACC	FFFFFF80h

Syntax **LBRK <constant>**

Operands 0 ≤ constant ≤ 255

Operation (PC) + 1 → PC
constant → BA0

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	0	1	0	1	0	0	0	X	X	X	Constant							

Description The BA0 register (base address register for DMEM0) is loaded with the specified 8–bit constant.

Category 3

Cycles 1 (Class 1)

Repeatable no

Example

```
LBRK 132
```

Before Instruction:		After Instruction:	
BA0	XXh	BA0	84h

Syntax **LCAA**

Operation (PC) + 1 → PC
$(AACC)_{31-24}$ → CA

Affected by SFAO.

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0
X X X X X X	0 0 0 1 0 0 0 0	X X X X X X X X X X

Description The output of the ALU accumulator (AACC) is shifted as specified by SFAO and the higher 8 bits are loaded to the CA register (coefficient memory address register).

For a description of how the SFAO control bit affects the operand, refer to the SFAO instruction description.

Category 2J

Cycles 1 (Class 1)

Repeatable yes

Example Given Conditions:SFAO=0.

```
LCAA
```

Before Instruction:		After Instruction:	
CA	XXh	CA	1Fh
AACC	1F000000h	AACC	1F000000h

Syntax **LCAC <constant>**

Operands 0 ≤ constant ≤ 255

Operation (PC) + 1 → PC
If (AACC) ≥ 0:
Then constant → CA
Else (CA) → CA

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0
1 1 1 1 1 1	1 0 0 0 0 0 0 0	X X X Constant

Description If the contents of the ALU accumulator (AACC) are greater than or equal to zero, the CA register (coefficient memory address register) is loaded with the specified 8-bit positive constant. Otherwise, the contents of the CA register remain unchanged.

Category 3

Cycles 1 (Class 7)

Repeatable no

Example 1

```
LCAC 136
```

Before Instruction:		After Instruction:	
AACC	F8735678h	AACC	F8735678h
CA	24h	CA	24h

Example 2

```
LCAC 136
```

Before Instruction:		After Instruction:	
AACC	01234567h	AACC	01234567h
CA	24h	CA	88h

Syntax **LCAK <constant>**

Operands 0 ≤ constant ≤ 255

Operation (PC) + 1 → PC
constant → CA

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11	10 9 8	7 6 5 4 3 2 1 0
1 1 1 1 1 1	0 0 1 1 0 0 0	X X X	Constant

Description The CA register (coefficient memory address register) is loaded with the specified 8–bit constant.

Category 3

Cycles 1 (Class 1)

Repeatable no

Example

```
LCAK 124
```

Before Instruction: CA [xxh]

After Instruction: CA [7Ch]

Syntax **LDPK <constant>**

Operands 0 ≤ constant ≤ 1

Operation (PC) + 1 → PC
constant → DBP

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11	10 9 8 7 6 5 4 3 2 1 0
1 1 1 1 1 1	1 0 0 0 1 0 K	X X X X X X X X X X X

Description DBP (data memory bank pointer) is loaded with a 1–bit constant. The value of DBP defines what data memory bank is accessed. When DBP=0, DMEM0 is accessed and when DBP=1, DMEM1 is accessed. The data memory addressing unit of the TMC57000, TMC57001 and TMS57002 is slightly different. Please refer to Section 3.3.3, Data Memory and Addressing, for more details.

The value of DBP is irrelevant with the ADDS, AMPY, AMAC, LD0T and STD1 instructions.

Category 2N

Cycles 1 (Class 1)

Repeatable no

Example

```
LDPK 1
```

Before Instruction: DBP [x]

After Instruction: DBP [1]

Syntax

Direct: **LDOT D(<dma>)**
Indirect: **LDOT {D*/D*+}**

Operands

0 ≤ dma ≤ 255
0 ≤ dma ≤ 31 (DMEM1 of TMS57002 only)

Operation

(PC) + 1 → PC
$(dma)_{23-0}$ → TREG

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	0	0	1	1	0	0	Memory Address. See Sections 4.2/4.3										

Description

TREG (temporary register) is loaded with the data of the addressed data memory location of data memory bank 0 (DMEM0)

The setting of DBP does not affect the execution of this instruction.

The LDOT instruction is used in conjunction with the STD1 instruction to transfer a data value from DMEM0 to DMEM1 during linear phase FIR filter calculations. Refer to Section 5.4 for more on linear phase FIR filter applications.

Note:

Take care to avoid a syntax error when using this instruction. The '0' in LDOT is the number 0 not the letter 'O'.

Category

2A

Cycles

1 (Class 1)

Repeatable

yes

Example

Given Conditions:BA0=0.

```
LDOT D(40)
or
LDOT D* ;ID=40.
```

	Before Instruction:	After Instruction:
DMEM0 40	123400h	123400h
TREG	XXXXXXh	123400h

Syntax

LIRA

Operands

None

Operation

(PC) + 1 → PC
$(AACC)_{31-24}$ → ID

Affected by SFAO.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	0	0	1	0	0	1	X	X	X	X	X	X	X	X	X	X	X

Description

The output of the ALU accumulator (AACC) is shifted as specified by SFAO; the ID register (data memory index register) is loaded with the higher 8 bits of this shifted AACC output value.

For a description of how the SFAO control bit affects the operand, refer to the SFAO instruction description.

Category

2J

Cycles

1 (Class 1)

Repeatable

yes

Example

Given Conditions:SFAO=0.

```
LIRA
```

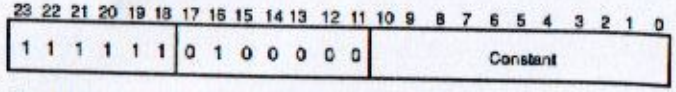
	Before Instruction:	After Instruction:
AACC	86000000h	86000000h
ID	XXh	86h

Syntax **LIRK <constant>**

Operands 0 ≤ constant ≤ 255

Operation (PC) + 1 → PC
constant → ID

Encoding



Description The ID register (data memory index register) is loaded with the specified 8-bit positive constant.

Category 3

Cycles 1 (Class 1)

Repeatable no

Example

```
LIRK 86
```

	Before Instruction:	After Instruction:
ID	15h	56h

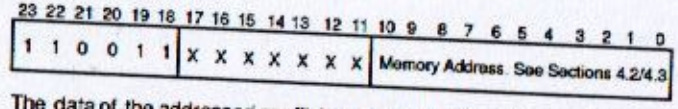
Syntax Direct: **LMHC C(<cma>)**
Indirect: **LMHC {C*/C*+}**

Operands 0 ≤ cma ≤ 255

Operation (PC) + 1 → PC
$(cma)_{31-0} \rightarrow MACC_{47-16}$
$(cma)_{31} \rightarrow MACC_{51-48}$
$0 \rightarrow MACC_{15-0}$

Affected by CRM.

Encoding



Description The data of the addressed coefficient memory location are provided by CRM and loaded into bits 47 through 16 of the MAC accumulator (MACC). Bits 51 through 48 of the MACC are sign extended and bits 15 through 0 are filled with zeroes.

If LMHC is executed following a multiplication instruction or MACC shift instruction, a pipeline conflict occurs. The LMHC instruction takes precedence and the previous instruction is cancelled.

Category 1C

Cycles 1 (Class 1)

Repeatable no

Example Given Conditions:CRM=0.

```
LMHC C(125)
or
LMHC C* ;CA=125.
```

	Before Instruction:	After Instruction:
CMEM 125	FF987654h	FF987654h
MACC	XXXXXXXXXXXXh	FF9876540000h

Syntax

Direct: **LMHD D(<dma>)**
Indirect: **LMHD {D*/D*+}**

Operands

0 ≤ dma ≤ 255
0 ≤ dma ≤ 31 (DMEM1 of TMS57002 only)

Operation

(PC) + 1 → PC
$(dma)_{23-0} \rightarrow MACC_{47-24}$
$(dma)_{23} \rightarrow MACC_{51-48}$
$0 \rightarrow MACC_{23-0}$

Affected by DBP.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	1	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed data memory location are loaded into bits 47 through 24 of the MAC accumulator (MACC). Bits 51 through 48 of the MACC are sign extended and bits 23 through 0 are filled with zeroes.

If LMHD is executed following a multiplication instruction or MACC shift instruction, a pipeline conflict occurs. The LMHD instruction takes precedence and the previous instruction is cancelled.

When executing LMHD and LMLD in succession, make sure to execute LMHD first because the LMHD instruction clears the lower order bits to zero.

Category

1A

Cycles

1 (Class 1)

Repeatable

no

Example

Given Conditions:BA0=0, DBP=0.

```
LMHD D(86)
or
LMHD D* ;ID=86.
```

Before Instruction:		After Instruction:	
DMEM0 86	004000h	DMEM0 86	004000h
MACC	XXXXXXXXXXXXh	MACC	004000000000h

Syntax

Direct: **LMLD D(<dma>)**
Indirect: **LMLD {D*/D*+}**

Operands

0 ≤ dma ≤ 255
0 ≤ dma ≤ 31 (DMEM1 of TMS57002 only)

Operation

(PC) + 1 → PC
$(dma)_{23-0} \rightarrow MACC_{23-0}$
$(MACC)_{51-24} \rightarrow MACC_{51-24}$

Affected by DBP.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed data memory location are loaded into bits 23 through 0 of the MACC. The remaining MACC bits remain unchanged.

If LMLD is executed following a multiplication instruction or MACC shift instruction, a pipeline conflict occurs. The LMLD instruction takes precedence and the previous instruction is cancelled.

When executing LMHD and LMLD in succession, make sure to execute LMHD first because the LMHD instruction clears the lower order bits to zero.

Category

1A

Cycles

1 (Class 1)

Repeatable

no

Example

Given Conditions:BA0=0, DBP=0.

```
LMLD D(86)
or
LMLD D* ;ID=86.
```

Before Instruction:		After Instruction:	
DMEM0 86	004000h	DMEM0 86	004000h
MACC	123456789012h	MACC	123456004000h

Syntax

Direct: **LPC C(<cma>)**
Indirect: **LPC {C*/C*+}**

Operands

$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(cma)_{31-24} \rightarrow HOST0$
$(cma)_{23-16} \rightarrow HOST1$
$(cma)_{15-8} \rightarrow HOST2$
$(cma)_{7-0} \rightarrow HOST3$

Affected by CRM.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	1	1	0	0	0	1	Memory Address. See Sections 4.2/4.3										

Description

The LPC instruction is available only for the TMS57002. The data of the addressed coefficient memory location are provided by CRM and loaded to the four 8–bit HOST registers.

This instruction is valid only when R/$\overline{W}$ is high. If $\overline{DREADY}$ is already low when LPC is executed, the LPC instruction is ignored.

Note:

With an assembler or simulator earlier than version 2.2., when using the LPC instruction, a "%" must be attached to the beginning of the instruction mnemonic (ie. %LPC).

For details on the host interface refer to Section 3.7.

Category

2C

Cycles

1 (Class 1)

Repeatable

no

Example

Given Conditions:CRM=0.

```
LPC C(20)
or
LPC C* ;CA=20.
```

	Before Instruction:	After Instruction:
CMEM 20	01234567h	01234567h
HOST0	XXh	01h
HOST1	XXh	23h
HOST2	XXh	45h
HOST3	XXh	67h

Syntax

Direct: **LPD D(<dma>)**
Indirect: **LPD {D*/D*+}**

Operands

$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(dma)_{23-16} \rightarrow HOST0$
$(dma)_{15-8} \rightarrow HOST1$
$(dma)_{7-0} \rightarrow HOST2$

Affected by DBP.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	1	1	0	0	0	0	Memory Address. See Sections 4.2/4.3										

Description

The LPD instruction is available only for the TMS57002. The data of the addressed data memory location are loaded to the first three of the four 8–bit HOST registers.

This instruction is valid only when R/$\overline{W}$ is high. If $\overline{DREADY}$ is already low when LPD is executed, the LPD instruction is ignored.

Note:

With an assembler or simulator earlier than version 2.2., when using the LPD instruction, a "%" must be attached to the beginning of the instruction mnemonic (ie. %LPD).

For details on the host interface refer to Section 3.7.

Category 2A

Cycles 1 (Class 1)

Repeatable no

Example

Given Conditions:DBP=0.

```
LPD D(20)
or
LPD D* ;ID=15.
```

	Before Instruction:	After Instruction:
DMEM0 15	012345h	012345h
HOST0	XXh	01h
HOST1	XXh	23h
HOST2	XXh	45h

Syntax

Direct: **MAC D(<dma>), {C*/C*+}**
MAC {D*/D*+}, C(<cma>)
Indirect: **MAC {D*/D*+}, {C*/C*+}**

Operands

$0 \le cma \le 255$, $0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(dma)_{23-0} \times (cma)_{31-0} + (MACC)_{51-0} \rightarrow MACC_{51-0}$

Affected by CRM, DBP, SFMA.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	0	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed coefficient memory location are provided by CRM, the output of the MAC accumulator (MACC) is shifted as specified by SFMA; the coefficient memory value is multiplied by the data of the addressed data memory location and this result is added to the shifted MACC output value. The result is stored in the MACC.

How the CRM, DBP and SFMA control bits affect the operands is described in the SCRM, LDPK and SFMA instruction descriptions, respectively.

Note:

Take care in choosing the instruction that follows an MAC(1) instruction. The MAC(1) execution is pipelined. This instruction requires one machine cycle but the result is not valid in time for the following instruction to use the result for calculation unless the MAC(1) instruction is followed by a MAC, MACS, MACU, AMAC, AMPY, SFML or SFMR instruction. For all other instructions, the result is valid in time for the second instruction from the MAC(1) instruction to use the result for calculation.

Category

1E

Cycles

1 (Class 3)

Repeatable

yes

Example

Given Conditions:BA0=0, CRM=0, DBP=0, SFAO=0, SFMA=0.

```
MAC D(86), C* ;CA=125.
or
MAC D*, C(125) ;ID=86.
or
MAC D*, C* ;CA=125.
```

	Before Instruction:	**After Instruction:**
DMEM0 86	000400h	000400h
CMEM 125	00012300h	00012300h
MACC	003153D306000h	003153009180Dh

Syntax

Direct: **MAC A, D(<dma>)**
Indirect: **MAC A, {D*/D*+}**

Operands

$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(AACC)_{31-0} \times (dma)_{23-0} + (MACC)_{51-0} \rightarrow MACC_{51-0}$

Affected by DBP, SFAO, SFMA.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	1	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The output of the ALU accumulator (AACC) is shifted as specified by SFAO, the output of the MAC accumulator (MACC) is shifted as specified by SFMA; the shifted AACC output value is multiplied by the data of the addressed data memory location and this result is added to the shifted MACC output value. The result is stored in the MACC.

How the DBP, SFAO and SFMA control bits affect the operands is described in the LDPK, SFAO and SFMA instruction descriptions, respectively.

Note:

Take care in choosing the instruction that follows an MAC(2) instruction. The MAC(2) execution is pipelined. This instruction requires one machine cycle but the result is not valid in time for the following instruction to use the result for calculation unless the MAC(2) instruction is followed by a MAC, MACS, MACU, AMAC, AMPY, SFML or SFMR instruction. For all other instructions, the result is valid in time for the second instruction from the MAC(2) instruction to use the result for calculation.

Category 1A

Cycles 1 (Class 3)

Repeatable yes

Example

Given Conditions:BA0=0, DBP=0, SFAO=0, SFMA=0.

```
MAC A, D(86)
or
MAC A, D* ;ID=86.
```

	Before Instruction:	After Instruction:
DMEM0 86	000020h	000020h
AACC	00012300h	00012300h
MACC	003153000000h	003153003480h

Syntax

Direct: **MAC C(<cma>), A**
Indirect: **MAC {C*/C*+}, A**

Operands

$0 \leq cma \leq 255$

Operation

$(PC) + 1 \rightarrow PC$

$(cma)_{31-0} \times (AACC)_{31-8} + (MACC)_{51-0} \rightarrow MACC_{51-0}$

Affected by CRM, SFAO, SFMA.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	0	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed coefficient memory location are provided by CRM, the output of the ALU accumulator (AACC) is shifted as specified by SFAO, the output of the MAC accumulator (MACC) is shifted as specified by SFMA; the coefficient memory value is multiplied by the shifted AACC output value and this result is added to the shifted MACC output value. The result is stored in the MACC.

How the CRM, SFAO and SFMA control bits affect the operands is described in the SCRM, SFAO and SFMA instruction descriptions, respectively.

Note:

Take care in choosing the instruction that follows an MAC(3) instruction. The MAC(3) execution is pipelined. This instruction requires one machine cycle but the result is not valid in time for the following instruction to use the result for calculation unless the MAC(3) instruction is followed by a MAC, MACS, MACU, AMAC, AMPY, SFML or SFMR instruction. For all other instructions, the result is valid in time for the second instruction from the MAC(3) instruction to use the result for calculation.

Category

1C

Cycles

1 (Class 3)

Repeatable

yes

Example

Given Conditions:CRM=0, SFAO=0, SFMA=0.

```
MAC C(125), A
or
MAC C*, A ;CA=125.
```

	Before Instruction:	After Instruction:
AACC	00005000h	00005000h
CMEM 125	00012300h	00012300h
MACC	003153000C000h	003153000B5E0h

Syntax

Direct: **MACS C(<cma>), A**
Indirect: **MACS {C*/C*+}, A**

Operands

$0 \le cma \le 255$

Operation

$(cma)_{31-0} \times (AACC)_{31-0} \times 2^{+1} + (MACC)_{51-0} \rightarrow MACC_{51-0}$

Affected by CRM, SFAO, SFMA.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	0	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed coefficient memory location are provided by CRM, the output of the ALU accumulator is shifted as specified by SFAO, the output of the MAC accumulator (MACC) is shifted as specified by SFMA; the higher 25 bits of this shifted AACC output value are shifted 1 bit left by SFMI and then multiplied by the coefficient memory value. This result is added to the shifted MACC output value. The result is stored in the MACC.

This instruction, in addition to the ADDS, AMAC, AMPY, LD0T and STD1 instructions, are very useful in implementing linear phase FIR filter calculations. Please refer to Section 5.4, Linear Phase FIR Filters, for more details.

Note:

Take care in choosing the instruction that follows an MACS instruction. The MACS execution is pipelined. This instruction requires one machine cycle but the result is not valid in time for the following instruction to use the result for calculation unless the MACS instruction is followed by a MAC, MACS, MACU, AMAC, AMPY, SFML or SFMR instruction. For all other instructions, the result is valid in time for the second instruction from the MACS instruction to use the result for calculation.

Category

1C

Cycles

1 (Class 3)

Repeatable

yes

Example

Given Conditions:CRM=0, SFAO=0, SFMA=0.

```
MACS C(125), A
or
MACS C*+, A ;CA=125.
```

	Before Instruction:	After Instruction:
AACC	00001000h	00001000h
CMEM 125	00012300h	00012300h
MACC	003153000000h	003153004800h

Syntax

Direct: **MACU D(<dma>), {C*/C*+}**
MACU {D*/D*+}, C(<cma>)
Indirect: **MACU {D*/D*+}, {C*/C*+}**

Operands

$0 \le cma \le 255$, $0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$unsigned(dma)_{23-0} \times (cma)_{31-0} + (MACC)_{51-0} \rightarrow MACC_{51-0}$

Affected by CRM, DBP, SFMA.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	1	x	x	x	x	x	x	x	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed coefficient memory location are provided by CRM, the output of the MAC accumulator (MACC) is shifted as specified by SFMA; the unsigned data of the addressed data memory location are multiplied by this coefficient memory value, and the result is added to the shifted MACC output value. The result is stored in the MACC.

How the CRM, DBP and SFMA control bits affect the operands is described in the SCRM, LDPK and SFMA instruction descriptions, respectively.

Note:

Take care in choosing the instruction that follows an MACU(1) instruction. The MACU(1) execution is pipelined. This instruction requires one machine cycle but the result is not valid in time for the following instruction to use the result for calculation unless the MACU(1) instruction is followed by a MAC, MACS, MACU, AMAC, AMPY, SFML or SFMR instruction. For all other instructions, the result is valid in time for the second instruction from the MACU(1) instruction to use the result for calculation.

Category

1E

Cycles

1 (Class 3)

Repeatable

yes

Example

Given Conditions:CRM=0, DBP=0, SFMA=0.

```
MACU D(86), C* ;CA=125.
or
MACU D*, C(125) ;ID=86.
or
MACU D*, C* ;CA=125, ID=86.
```

	Before Instruction:	After Instruction:
CMEM 125	00012300h	00012300h
DMEM0 86	800000h	800000h
MACC	003100000000h	003123000000h

Syntax

Direct: **MACU A, D(<dma>)**
Indirect: **MACU A, {D*/D*+}**

Operands

$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(AACC)_{31-0} \times unsigned(dma)_{31-0} + (MACC)_{51-0} \rightarrow MACC_{51-0}$

Affected by DBP, SFAO, SFMA.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	0	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The output of the ALU accumulator (AACC) is shifted as specified by SFAO, the output of the MAC accumulator (MACC) is shifted as specified by SFMA, the unsigned data of the addressed data memory location are multiplied by this shifted AACC output value, and this result is added to the shifted MACC output value. The result is stored in the MACC.

How the DBP, SFAO and SFMA control bits affect the operands is described in the LDPK, SFAO and SFMA instruction descriptions, respectively.

Note:

Take care in choosing the instruction that follows an MACU(2) instruction. The MACU(2) execution is pipelined. This instruction requires one machine cycle but the result is not valid in time for the following instruction to use the result for calculation unless the MACU(2) instruction is followed by a MAC, MACS, MACU, AMAC, AMPY, SFML or SFMR instruction. For all other instructions, the result is valid in time for the second instruction from the MACU(2) instruction to use the result for calculation.

Category

1A

Cycles

1 (Class 3)

Repeatable

yes

Example

Given Conditions:BA0=0, DBP=0, SFAO=0, SFMA=0.

```
MACU A, D(86)
or
MACU A, D* ;ID=86.
```

	Before Instruction:	After Instruction:
AACC	00012300h	00012300h
DMEM0 86	800000h	800000h
MACC	018000000000h	018012300000h

Syntax

Direct: **MPY D{<dma>}, {C*/C*+}**
MPY {D*/D*+}, C{<cma>}
Indirect: **MPY {D*/D*+}, {C*/C*+}**

Operands

$0 \le cma \le 255$, $0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(dma)_{23-0} \times (cma)_{31-0} \rightarrow MACC_{51-0}$

Affected by CRM, DBP.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3											

Description

The data of the addressed coefficient memory location are provided by CRM; this coefficient memory value is multiplied by the data of the addressed data memory location. The result is stored in the MAC accumulator (MACC).

How the CRM and DBP control bits affect the operands is described in the SCRM and LDPK instruction descriptions, respectively.

Note:

Take care in choosing the instruction that follows an MPY(1) instruction. The MPY(1) execution is pipelined. This instruction requires one machine cycle but the result is not valid in time for the following instruction to use the result for calculation unless the MPY(1) instruction is followed by a MAC, MACS, MACU, AMAC, AMPY, SFML or SFMR instruction. For all other instructions, the result is valid in time for the second instruction from the MPY(1) instruction to use the result for calculation.

Category 1E

Cycles 1 (Class 3)

Repeatable yes

Example

Given Conditions:BA0=0, CRM=0, DBP=0.

```
MPY D(86), C* ;CA=125.
or
MPY D*, C(125) ;ID=86.
or
MPY D*, C* ;CA=125.
```

	Before Instruction:	After Instruction:
DMEM0 86	000400h	000400h
CMEM 125	00012300h	00012300h
MACC	XXXXXXXXXXXXh	000000091800h

Syntax

Direct: **MPY C(<cma>), A**
Indirect: **MPY [C*/C*+], A**

Operands

$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(cma)_{31-0} \times (AACC)_{31-0} \rightarrow MACC_{51-0}$

Affected by CRM, SFAO.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The output of the ALU accumulator (AACC) is shifted as specified by SFAO; this shifted AACC output value is multiplied by the data of the addressed data memory location. The result is stored in the MACC.

How the CRM and SFAO control bits affect the operands is described in the SCRM and SFAO instruction descriptions, respectively.

Note:

Take care in choosing the instruction that follows an MPY(2) instruction. The MPY(2) execution is pipelined. This instruction requires one machine cycle but the result is not valid in time for the following instruction to use the result for calculation unless the MPY(2) instruction is followed by a MAC, MACS, MACU, AMAC, AMPY, SFML or SFMR instruction. For all other instructions, the result is valid in time for the second instruction from the MPY(2) instruction to use the result for calculation.

Category 1C

Cycles 1 (Class 3)

Repeatable yes

Example

Given Conditions:CRM=0, SFAO=0.

```
MPY C(125), A
or
MPY C*, A ;CA=125.
```

	Before Instruction:	After Instruction:
AACC	00002000h	00002000h
CMEM 125	00012300h	00012300h
MACC	003153000000h	000000004800h

Syntax

Direct: **MPY CREG, D(<dma>)**
Indirect: **MPY CREG, {D*/D*+}**

Operands

$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(CREG) \times (dma)_{23-0} \rightarrow MACC_{51-0}$

Affected by DBP.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed data memory location are multiplied by the contents of CREG. The result is stored in the MAC accumulator (MACC).

CREG contains the value which was set by the last instruction which used the MACC.

Note:

Take care in choosing the instruction that follows an MPY(3) instruction. The MPY(3) execution is pipelined. This instruction requires one machine cycle but the result is not valid in time for the following instruction to use the result for calculation unless the MPY(3) instruction is followed by a MAC, MACS, MACU, AMAC, AMPY, SFML or SFMR instruction. For all other instructions, the result is valid in time for the second instruction from the MPY(3) instruction to use the result for calculation.

Category 1A

Cycles 1 (Class 3)

Repeatable yes

Example

Given Conditions:BA0=0, DBP=0.

```
MPY CREG, D(86)
or
MPY CREG, D* ;ID=86.
```

	Before Instruction:	After Instruction:
$DMEM0_{86}$	000020h	000020h
CREG	00012300h	00012300h
MACC	XXXXXXXXXXXXh	000000048C0h

Syntax

Direct: **MPYU D(<dma>), {C*/C*+}**
MPYU {D*/D*+}, C(<cma>)
Indirect: **MPYU {D*/D*+}, {C*/C*+}**

Operands

$0 \le cma \le 255$, $0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$unsigned(dma)_{23-0} \times (cma)_{31-0} \rightarrow MACC_{51-0}$

Affected by CRM, DBP.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	0	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed coefficient memory location are provided by CRM; this coefficient memory value is multiplied by the unsigned data of the addressed data memory location. The result is stored in the MAC Accumulator (MACC).

How the CRM and DBP control bits affect the operands is described in the SCRM and LDPK instruction descriptions, respectively.

> **Note:**
>
> Take care in choosing the instruction that follows an MPYU instruction. The MPYU execution is pipelined. This instruction requires one machine cycle but the result is not valid in time for the following instruction to use the result for calculation unless the MPYU instruction is followed by a MAC, MACS, MACU, AMAC, AMPY, SFML or SFMR instruction. For all other instructions, the result is valid in time for the second instruction from the MPYU instruction to use the result for calculation.

Category 1E

Cycles 1 (Class 3)

Repeatable yes

Example

Given Conditions:CRM=0, DBP=0.

```
MPYU D(86), C* ;CA=125.
or
MPYU D*, C(125) ;ID=86.
or
MPYU D*, C* ;CA=125, ID=86.
```

	Before Instruction:	**After Instruction:**
CMEM 125	00012300h	00012300h
DMEM0 86	800000h	800000h
MACC	xxxxxxxxxxxxh	000123000000h

Syntax NEG

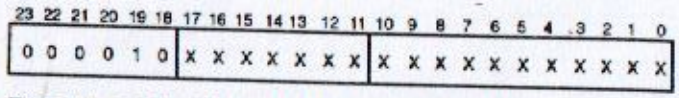
Operands None

Operation

(PC) + 1 → PC
(AACC) x (−1) → AACC

Affected by SFAO.
Affects AOV.

Encoding



Description

The output of the ALU accumulator (AACC) is shifted as specified by SFAO; the contents of the ALU accumulator (AACC) are replaced with the arithmetic complement (twos–complement) of the shifted AACC output value.

Note that if after shifting the AACC output as specified by SFAO, the result is 80000000h, this is a special case. The NEG of 80000000h is 80000000h and AOV (AACC overflow flag) is set to one. AOV is cleared to zero at synchronization, reset or with the BV, INCD or RAOV instructions. For a description of how the SFAO control bit affects the operand, refer to the SFAO instruction description.

Category 1I

Cycles 1 (Class 2)

Repeatable no

Example 1

```
NEG ;SFAO=0
```

Before Instruction:			After Instruction:		
AACC	FFFFF228h	AOV X	AACC	00000DD8h	AOV X

Example 2

```
NEG ;SFAO=0
```

Before Instruction:			After Instruction:		
AACC	80000000h	AOV X	AACC	80000000h	AOV X

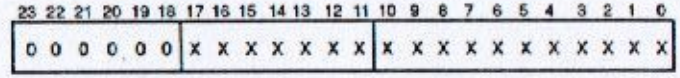
Syntax NOP1

Operands None

Operation

If NOP1 is followed by a secondary instruction,
  Then execute the secondary instruction.
If NOP1 is followed by NOP2 or a blank field,
  Then (PC) + 1 → PC.

Encoding



Description No primary instruction is performed. If no instruction is specified in the primary instruction field, NOP1 is automatically generated; bits 23–18 are zero–filled by the 5700x assembler.

Category 1N

Cycles 1 (Class 1)

Repeatable yes

Example

```
NOP1  ;No primary instruction is performed.
```

Syntax **NOP2**

Operands None

Operation If NOP2 is preceded by a primary instruction,
Then execute the primary instruction.
If NOP2 is preceded by NOP1 or a blank field,
Then (PC) + 1 → PC.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X

Description No secondary instruction operation is performed. If no instruction is specified in the secondary instruction field, NOP2 is automatically generated; bits 17–11 are zero–filled by the 5700x assembler.

Category 2N

Cycles 1 (Class 1)

Repeatable yes

Example

```
NOP2  ;No secondary instruction is performed.
```


Syntax

Direct: **OR D(<dma>), A**
Indirect: **OR {D*/D*+}, A**

Operands

0 ≤ dma ≤ 255
0 ≤ dma ≤ 31 (DMEM1 of TMS57002 only)

Operation

(PC) + 1 → PC
$(dma)_{23-0}$ OR $(AACC)_{31-8}$ → $AACC_{31-8}$
$(AACC)_{7-0}$ → $AACC_{7-0}$

Affected by DBP, SFAI, SFAO.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	1	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description The data of the addressed data memory location are right–shifted as specified by SFAI, the output of the ALU accumulator (AACC) is shifted as specified by SFAO; these two values are ORed bitwise and the result is stored in the AACC. Note that the lower 8 bits of the shifted AACC output value are ORed bitwise with all zeroes.

How the DBP, SFAI and SFAO control bits affect the operands is described in the LDPK, SFAI and SFAO instruction descriptions, respectively.

Category 1A

Cycles 1 (Class 2)

Repeatable no

Example Given Conditions:BA0=0, DBP=0, SFAI=0, SFAO=0.

```
OR D(51), A
or
OR D*, A ;ID=51.
```

Before Instruction:

DMEM0 51: 000F00h

AACC: 120030F0h

After Instruction:

DMEM0 51: 000F00h

AACC: 120F30F0h

Syntax

Direct: **OR C(<cma>), A**
Indirect: **OR {C*/C*+}, A**

Operands

$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(cma)_{31-0}$ OR $(AACC)_{31-0} \rightarrow AACC_{31-0}$

Affected by CRM, SFAO.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed coefficient memory location are provided by CRM, the output of the ALU accumulator (AACC) is shifted as specified by SFAO; these two values are ORed bitwise and the result is stored in the AACC.

How the CRM and SFAO control bits affect the operands is described in the SCRM and SFAO instruction descriptions, respectively.

Category

1C

Cycles

1 (Class 2)

Repeatable

no

Example

Given Conditions:CRM=0, SFAO=0.

```
OR C(187), A
or
OR C*, A ;ID=187.
```

	Before Instruction:	After Instruction:
CMEM 187	000DFFFFh	000DFFFFh
AACC	12345678h	1234FFFFh

Syntax

Direct: **OR D(<dma>), {C*/C*+}**
OR {D*/D*+}, C(<cma>)
Indirect: **OR {D*/D*+}, {C*/C*+}**

Operands

$0 \le cma \le 255$, $0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(dma)_{23-0}$ OR $(cma)_{31-8} \rightarrow AACC_{31-8}$
$(cma)_{7-0} \rightarrow AACC_{7-0}$

Affected by CRM, DBP, SFAI.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	1	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed data memory location are right-shifted as specified by SFAI, the data of the addressed coefficient memory location are provided by CRM; these two values are ORed bitwise and the result is stored in the ALU accumulator (AACC). Note that the lower 8 bits of the coefficient memory value are ORed bitwise with all zeroes.

How the CRM and SFAI control bits affect the operands is described in the SCRM and SFAI instruction descriptions, respectively.

Category

1E

Cycles

1 (Class 2)

Repeatable

no

Example

Given Conditions:BA0=0, CRM=0, DBP=0, SFAI=0.

```
OR D(86), C* ;CA=125.
or
OR D*, C(125) ;ID=86.
or
OR D*, C* ;CA=125, ID=86.
```

	Before Instruction:	After Instruction:
DMEM0 86	000FFFh	000FFFh
CMEM 125	12345678h	12345678h
AACC	XXXXXXXXh	123FFF78h

Syntax **RAOV**

Operands None

Operation (PC) + 1 → PC
0 → AOV

Affects AOV.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	1	1	1	0	0	0	X	X	X	X	X	X	X	X	X	X	X

Description The RAOV instruction resets the AOV (AACC overflow flag) control bit to zero. AOV is also reset at synchronization, reset or with the BV or INCD instructions.

The state of AOV is output at the $\overline{\text{OVFA}}$ pin.

Category 2N

Cycles 1 (Class 1)

Repeatable no

Example RAOV

Before Instruction:	After Instruction:
AOV [1]	AOV [0]

Syntax Direct: **RDE C(<cma>)**
Indirect: **RDE {C*/C*+}**

Operands 0 ≤ cma ≤ 255

Operation (PC) + 1 → PC
$(cma)_{18-0}$ → XOA
access address = XOA + XBA
external RAM data → XRD

Affected by CRM, SEL, SRAM, WORD.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	1	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description This instruction is available only for the TMS57002.

The data of the addressed coefficient memory location are provided by CRM, and the lower 19 bits of this value are loaded to the external RAM interface offset address register (XOA). One word of data is read from external memory and stored in the read data buffer (XRD).

The sum of XOA and XBA (external RAM interface base address register) are used in determining the access address into external memory. See Section 3.9.5 for more on external memory addressing.

The number of machine cycles required to read one word from external memory and load it to the XRD buffer is provided in Table 3–18 and Table 3–19. If an RDE, WRE or REF instruction is executed during this required number of machine cycles, the RDE, WRE or REF instruction will be ignored. The value of the SEL, SRAM and WORD control bits define the required number of machine cycles (see Table 3–15).

In the case of 16–bit data (WORD=0), the lower 8 bits of the XRD buffer are filled with zero.

Category 1C

Cycles 1 (Class 1)

Repeatable no

Example Given Conditions:CRM=0, SEL=1, SRAM=0,WORD=0.

```
RDE C(125)
or
RDE C* ;CA=125.
```

	Before Instruction:	After Instruction:
CMEM 125	00000001h	00000001h
XBA	0008h	0008h
XOA	XXXXh	0001h
XRD	XXXXXXh	123400h

	Before Write Access:	After Write Access Complete:
External RAM	12h	12h
	34h	34h

Syntax **REF**

Operands None

Operation (PC) + 1 → PC

Affected by SRAM.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	0	0	1	1	1	0	X	X	X	X	X	X	X	X	X	X	X

Description The REF instruction is available only for the TMS57002.

CAS–before–RAS refresh is provided with this instruction. The refresh operation requires three machine cycles, one cycle to trigger and two additional cycles to complete the operation. Therefore, the two instructions following an REF instruction should not include an RDE, WRE or REF instruction. If either of the two instructions following an REF instruction include an RDE, WRE or REF instruction, they will be ignored.

Note that when using DRAM, the SRAM control bit must be set to zero.

Category 2N

Cycles 1 (Class 1)

Repeatable no

Syntax **RMOM**

Operands None

Operation (PC) + 1 → PC
0 → MOVM

Affects MOVM.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	1	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X

Description The RMOM instruction resets the MOVM (MACC overflow mode) control bit to zero, which disables the overflow mode at the MACC output. When MOVM=0, if overflow is detected at the MACC (MOV=1), the overflowed value will be outputted.

When overflow occurs, clipping of the MACC output is available if MOVM is set to one. MOVM is set to one with the SMOM instruction. Please refer to the SMOM instruction for more details.

Category 2N

Cycles 1 (Class 1)

Repeatable no

Example

```
RMOM
```

Before Instruction:	After Instruction:
MOVM [1]	MOVM [0]

Syntax **RMOV**

Operands None

Operation (PC) + 1 → PC
0 → MOV

Affects MOVM.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	1	1	1	0	1	0	X	X	X	X	X	X	X	X	X	X	X

Description The RMOV instruction resets the MOV (MACC overflow flag) control bit to zero. MOV is also reset by the execution of the INCD instruction and at synchronization and reset.

The state of MOV is output at the $\overline{\text{OVFM}}$ pin.

Category 2N

Cycles 1 (Class 1)

Repeatable no

Example

```
RMOV
```

Before Instruction:	After Instruction:
MOV [1]	MOV [0]

Syntax

RND <constant>

Operands

constant = 0, 16, 20, 24, 32 or 48

Operation

(PC) + 1 → PC
mapping(constant) → RND

Affects RND.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	1	1	0	1	K	K	K	X	X	X	X	X	X	X	X	X	X	X

Description

The RND instruction sets the RND (MACC output rounding) control bit as defined by the specified constant as shown in the table below. The MAC accumulator (MACC) output, after passing through the Output Shifter, is rounded in the Output Rounder as defined by the value of RND as shown below. An overflow condition is checked for after rounding is performed.

Constant	RND control bit	Rounding Mode of MACC Output Rounder
0 or 48	0	No rounding
32	1	Round to a 32–bit value. Lower 16 bits filled with 0.
24	2	Round to a 24–bit value. Lower 24 bits filled with 0.
20	3	Round to a 20–bit value. Lower 28 bits filled with 0.
16	4	Round to a 16–bit value. Lower 32 bits filled with 0.

Category

2N

Cycles

1 (Class 1)

Repeatable

no

Example

```
RND 24
```

Before Instruction:

RND [0]

After Instruction:

RND [2]

Syntax

RPTK <constant>

Operands

0 ≤ constant ≤ 255

Operation

(PC) + 1 → PC
constant → RPTC

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	0	0	1	0	0	0	0	X	X	X	K	K	K	K	K	K	K	K

Description

The 8–bit specified constant is loaded into the repeat counter (RPTC). This causes the instruction following the RPTK instruction to be executed one more time than the number loaded into the RPTC, provided that it is a repeatable instruction.

Whether or not an instruction is repeatable is indicated in the instruction description of each instruction. If the RPTK instruction is followed by an instruction which is not repeatable, the repeat mode is cancelled by clearing RPTC to zero. If the instruction following an RPTK instruction contains both a primary and secondary instruction, both must be repeatable. If both are not repeatable, the repeat mode is cancelled by clearing RPTC to zero. The assembler will warn the user if a non–repeatable instruction follows the RPTK instruction.

Category

3

Cycles

1 (Class 6)

Repeatable

no

Example

```
RPTK 4
ADD D*+, A* ;This instruction is executed 5 times.
```

Syntax

Direct: **SACC C(<cma>)**
Indirect: **SACC (C*/C*+)**

Operands

$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(AACC)_{31-0} \rightarrow cma_{31-0}$

Affected by SFAO.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	0	0	0	0	0	1	Memory Address. See Sections 4.2/4.3										

Description

The output of the ALU accumulator (AACC) is right–shifted as specified by SFAO; this shifted value is stored in the addressed coefficient memory location.

How the SFAO control bit affects the operands is described in the SFAO instruction description.

Category

2D

Cycles

1 (Class 1)

Repeatable

yes

Example

Given Conditions:SFAO=0.

```
SACC C(125)
or
SACC C* ;CA=125.
```

	Before Instruction:	After Instruction:
CMEM 125	XXXXXXXXh	01234567h
AACC	01234567h	01234567h

Syntax

Direct: **SACD D(<dma>)**
Indirect: **SACD (D*/D*+)**

Operands

$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(AACC)_{31-8} \rightarrow dma_{23-0}$

Affected by DBP, SFAO.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	0	0	0	0	1	0	Memory Address. See Sections 4.2/4.3										

Description

The output of the ALU accumulator (AACC) is right–shifted as specified by SFAO; the higher 24 bits of this shifted value are stored in the addressed data memory location.

How the SFAO control bit affects the operands is described in the SFAO instruction description.

Category

2B

Cycles

1 (Class 1)

Repeatable

yes

Example

Given Conditions:BA0=0, DBP=0, SFAO=0.

```
SACD D(86)
or
SACD D* ;ID=86.
```

	Before Instruction:	After Instruction:
DMEM0 86	000123h	0F3456h
AACC	0F345678h	0F345678h

Syntax

SCRM <constant>

Operands

$0 \leq \text{constant} \leq 2$

Operation

(PC) + 1 → PC
mapping(constant) → CRM

Affects CRM.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	1	0	0	1	0	K	K	X	X	X	X	X	X	X	X	X	X	X

Description

The SCRM instruction sets the CRM (CMEM read mode) control bit as defined by the specified constant as shown in the table below. The CRM control bit controls the format in which one word is read from coefficient memory and outputted to the C–Bus.

Constant	CRM control bit	Output of Coefficient Memory to C–Bus
0	0	$(cma)_{31-0} \rightarrow C\text{–}Bus_{31-0}$
1	1	$(cma)_{31-16} \rightarrow C\text{–}Bus_{31-16}$ $0 \rightarrow C\text{–}Bus_{15-0}$
2	2	$(cma)_{15-0} \rightarrow C\text{–}Bus_{31-16}$ $0 \rightarrow C\text{–}Bus_{15-0}$

Note: If a coefficient memory word can be represented by 16 bits, CMEM can be effectively expanded.

During CMEM update, CRM is by default equal to zero. See Section 3.7.3, Coefficient Memory Update, for more details.

Category

2N

Cycles

1 (Class 1)

Repeatable

no

Example

```
SCRM 1
```

Before Instruction:

CRM [0]

After Instruction:

CRM [1]

Syntax

SFAI <constant>

Operands

constant = 0 or –1

Operation

(PC) + 1 → PC
mapping(constant) → SFAI

Affects SFAI.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	1	0	1	0	1	0	K	X	X	X	X	X	X	X	X	X	X	X

Description

The SFAI instruction sets the SFAI control bit as defined by the specified constant according to the table below. The SFAI control bit controls the two 24–bit ALU Input Shifters.

If SFAI=0, there is no shifting and the value is extended by eight zero–filled bits to create a 32–bit input value to the ALU. If SFAI=1, 24–bit input from the A–Bus and D–Bus is shifted 1 bit to the right with the sign bit extended. This value is then extended by seven zero–filled bits.

Constant	SFAI control bit	Shifting Mode of AACC Input Shifter
0	0	No shifting.
–1	1	1–bit right arithmetic shift with sign extension (24–bit inputs only).

This input shifting function is especially useful in implementing linear phase FIR filter applications to avoid overflow during pre–addition of two data samples. Refer to Section 5.4, Linear Phase FIR Filters, for more details.

Category

2N

Cycles

1 (Class 1)

Repeatable

no

Example

```
SFAI -1
```

Before Instruction:

SFAI [0]

After Instruction:

SFAI [1]

Syntax **SFAO <constant>**

Operands constant = 0 or 7

Operation (PC) + 1 → PC
mapping(constant) → SFAO

Affects SFAO.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	1	0	1	0	0	0	K	X	X	X	X	X	X	X	X	X	X	X

Description The SFAO instruction sets the SFAO control bit as defined by the specified constant according to the table below. The SFAO control bit controls the Output Shifter of the ALU.

If SFAO=0, the output from the ALU accumulator (AACC) is not shifted. If SFAO=1, the AACC output is shifted 7 bit to the left and the lower 7 bit are filled with zero.

Constant	SFAO control bit	Shifting Mode of AACC Output Shifter
0	0	No shifting.
7	1	7–bit left logical shift. The 7 lower bits are filled with 0.

Category 2N

Cycles 1 (Class 1)

Repeatable no

Example

```
SFAO 7
```

Before Instruction: SFAO [0]

After Instruction: SFAO [1]

Syntax **SFMA <constant>**

Operands constant = 0, 2, 4 or –16

Operation (PC) + 1 → PC
mapping(constant) → SFMA

Affects SFMA.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	1	0	1	1	0	K	K	X	X	X	X	X	X	X	X	X	X	X

Description The SFMA instruction sets the SFMA control bit as defined by the specified constant according to the table below. The SFMA control bit controls the MAC Accumulation Shifter. The MAC Accumulation Shifter shifts the MAC accumulator output value as specified by SFMA before accumulation.

Constant	SFMA control bit	Result of Shifting at MAC Accumulation Shifter: Input to MAC Accumulation Shifter → Output from MAC Accumulation Shifter (AFTER_SHIFT)
0	0	$(MACC)_{51-0}$ → $AFTER\_SHIFT_{51-0}$
2	1	$(MACC)_{51}$ → $AFTER\_SHIFT_{51}$ $(MACC)_{48-0}$ → $AFTER\_SHIFT_{50-2}$ 0 → $AFTER\_SHIFT_{1-0}$
4	2	$(MACC)_{51}$ → $AFTER\_SHIFT_{50}$ $(MACC)_{45-0}$ → $AFTER\_SHIFT_{50-4}$ 0 → $AFTER\_SHIFT_{3-0}$
–16	3	$(MACC)_{51}$ → $AFTER\_SHIFT_{51-36}$ $(MACC)_{51-16}$ → $AFTER\_SHIFT_{35-0}$

Category 2N

Cycles 1 (Class 1)

Repeatable no

Example

```
SFMA 4
```

Before Instruction: SFMA [0]

After Instruction: SFMA [2]

Syntax SFML

Operands None

Operation

$(PC) + 1 \rightarrow PC$
$0 \rightarrow MACC_0$
$(MACC)_{49-0} \rightarrow MACC_{50-1}$
$(MACC)_{51} \rightarrow MACC_{51}$

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Description The SFML instruction performs a 1 bit arithmetic left shift on the contents of the MAC accumulator (MACC). The sign bit is copied into the MSB of the MACC and the LSB is filled with zero.

Note:

Take care in choosing the instruction that follows an SFML instruction. The SFML execution is pipelined. This instruction requires one machine cycle but the result is not valid in time for the following instruction to use the result for calculation unless the SFML instruction is followed by a MAC, MACS, MACU, AMAC, AMPY, SFML or SFMR instruction. For all other instructions, the result is valid in time for the second instruction from the SFML instruction to use the result for calculation.

Category 1N

Cycles 1 (Class 3)

Repeatable yes

Example

```
SFML
```

Before Instruction:		After Instruction:	
MACC	01234567890123h	MACC	02468ACF12024h

Syntax SFMO <constant>

Operands constant = 0, 2, 4 or –8

Operation

$(PC) + 1 \rightarrow PC$
mapping(constant) → SFMO

Affects SFMO.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	1	1	0	0	0	K	K	X	X	X	X	X	X	X	X	X	X	X

Description The SFMO instruction sets the SFMO (MACC output shifter) control bit as defined by the specified constant according to the table below. The output from the MAC accumulator (MACC) passes through the Output Shifter and is shifted according to the value of SFMO as shown below.

Constant	SFMA control bit	Result of Shifting at MAC Output Shifter: Input to MACC Output Shifter → Output from MACC Output Shifter (POST_SHIFT)
0	0	$(MACC)_{51-0} \rightarrow POST\_SHIFT_{51-0}$
2	1	$(MACC_{51} \rightarrow POST\_SHIFT_{51}$ $(MACC)_{48-0} \rightarrow POST\_SHIFT_{50-2}$ $0 \rightarrow POST\_SHIFT_{1-0}$
4	2	$(MACC)_{51} \rightarrow POST\_SHIFT_{51}$ $(MACC)_{46-0} \rightarrow POST\_SHIFT_{50-4}$ $0 \rightarrow POST\_SHIFT_{3-0}$
–8	3	$(MACC)_{51} \rightarrow AFTER\_SHIFT_{51-44}$ $(MACC)_{51-8} \rightarrow AFTER\_SHIFT_{43-0}$

Category 2N

Cycles 1 (Class 1)

Repeatable no

Example

```
SFMO 2
```

Before Instruction:		After Instruction:	
SFMO	0	SFMO	1

Syntax **SFMR**

Operands None

Operation
$(PC) + 1 \rightarrow PC$
$(MACC)_{51-1} \rightarrow MACC_{50-0}$
$(MACC)_{51} \rightarrow MACC_{51}$

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Description The SFMR instruction performs a 1 bit arithmetic right shift on the contents of the MAC accumulator (MACC). The sign bit is copied into the MSB of the MACC. Bit zero is lost.

Note:

Take care in choosing the instruction that follows an SFMR instruction. The SFMR execution is pipelined. This instruction requires one machine cycle but the result is not valid in time for the following instruction to use the result for calculation unless the SFMR instruction is followed by a MAC, MACS, MACU, AMPY, AMAC, SFML or SFMR instruction. for all other instructions, the result is valid in time for the second instruction from the SFMR instruction to use result for calculation.

Category 1N

Cycles 1 (Class 3)

Repeatable yes

Example SFMR

	Before Instruction:	After Instruction:
MACC	8012340000001h	C009IA0000000h

Syntax Direct: **SLMH D(<dma>)**
Indirect: **SLMH (D*/D*+)**

Operands $0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation
$(PC) + 1 \rightarrow PC$
$(MACC)_{47-32} \rightarrow dma_{23-8}$
$0 \rightarrow dma_{7-0}$

Affected by MOVM:
Affects MOV, SFMO only affects MOV.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	0	0	0	1	1	0	Memory Address. See Sections 4.2/4.3										

Description The higher 16 bits of the MAC accumulator (MACC), excluding the 4–bit head margin, are stored in the addressed data memory location with the lower 8 bits filled with zeroes.

Although the MACC output is not affected by the Output Shifter and the Output Rounder, overflow detection is performed. The number of MACC bits checked for overflow is defined by the value of SFMO. Refer to the SFMO instruction for further details.

In the case that overflow is detected (MOV=1) and MOVM=1, depending upon the direction of overflow, the value is clipped to either 7FFFFFFFFFFh or 8000000000000h and bits 47–32 are stored in the addressed data memory location instead of the MACC output value. If MOVM=0, the MACC output value is not clipped. Note that with this instruction, there is no rounding of a clipped result. See the SMOM instruction description for more details.

The SLMH instruction is useful for storing the result of a high precision multiplication calculation (40–bit DMEM x 32–bit CMEM) for an IIR filter application. Refer to Section 5.7 for more details.

Category 2B

Cycles 1 (Class 1)

Repeatable yes

Example

Given Conditions:BA0=0, DBP=0, MOVM=0, SFMO=0.

```
SLMH D(85)
or
SLMH D*+ ;ID=86.
```

Before Instruction:		After Instruction:	
DMEM 86	XXXXXXh	DMEM 86	123400h
MACC	0123456789012h	MACC	0123456789012h

Syntax

Direct: **SLML D(<dma>)**
Indirect: **SLML {D*/D*+}**

Operands

$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(MACC)_{31-8} \rightarrow dma_{23-0}$

Affected by MOVM;
Affects MOV; SFMO only affects MOV.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	0	0	0	0	1	1	1	Memory Address. See Sections 4.2/4.3										

Description

Bits 31 through 8 of the 52–bit MAC accumulator (MACC) are stored in the addressed data memory location.

Although the MACC output is not affected by the Output Shifter and the Output Rounder, overflow detection is performed. The number of MACC bits checked for overflow is defined by the value of SFMO. Refer to the SFMO instruction for further details.

In the case that overflow is detected (MOV=1) and MOVM=1, depending upon the direction of overflow, the value is clipped to either 7FFFFFFFFFFFh or 800000000000h and bits 31–8 of this clipped value are stored in the addressed data memory location instead of the MACC output value. If MOVM=0, the MACC output value is not clipped. Note that with this instruction, there is no rounding of a clipped result. See the SMOM instruction description for more details.

The SLML instruction is useful for storing the result of a high precision multiplication calculation (40–bit DMEM x 32–bit CMEM) for an IIR filter application. Refer to Section 5.7 for more details.

Category 2B

Cycles 1 (Class 1)

Repeatable yes

Example

Given Conditions:BA0=0, DBP=0, MOVM=0, SFMO=0.

```
SLMH D(87)
or
SLMH D*+ ;ID=87.
```

	Before Instruction:	After Instruction:
DMEM 87	XXXXXXh	567890h
MACC	0123456789012h	0123456789012h

Syntax

Direct: **SMHC C(<cma>)**
Indirect: **SMHC (C*/C*+)**

Operands

$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(MACC)_{47-16} \rightarrow cma_{31-0}$

Affected by MOVM, RND, SFMO.
Affects MOV.

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0
X X X X X X	0 0 0 0 1 0 1	Memory Address. See Sections 4.2/4.3

Description

The output of the MAC accumulator (MACC) is shifted and rounded as specified by SFMO and RND, respectively; the higher 32 bits of this value are stored in the addressed coefficient memory location. Refer to the SFMO and RND instruction descriptions for more details on how these control bits affect the operand.

The MACC output value is checked for overflow twice, after it passes through both the Output Shifter and the Output Rounder. In the case that overflow is detected (MOV=1) and MOVM=1, depending upon the direction of overflow, the value is clipped to either 7FFFFFFFFFFFh or 800000000000h, rounded as specified by RND and bits 47–16 are stored in the addressed coefficient memory location instead of the MACC output value. If MOVM=0, the MACC output value is not clipped.

Category

2D

Cycles

1 (Class 1)

Repeatable

yes

Example

Given Conditions:MOVM=1, RND=1, SFMO=0.

```
SMHC C(125)
or
SMHC C* ;CA=125.
```

	Before Instruction:	After Instruction:
CMEM 125	XXXXXXXXh	12345678h
MACC	0123456789012h	0123456789012h

Syntax

Direct: **SMHD D(<dma>)**
Indirect: **SMHD {D*/D*+}**

Operands

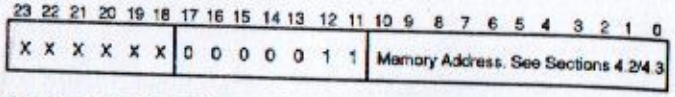
0 ≤ dma ≤ 255
0 ≤ dma ≤ 31 (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(MACC)_{47-24} \rightarrow dma_{23-0}$

Affected by MOVM, RND, SFMO.
Affects MOV.

Encoding



Description

The output of the MAC accumulator (MACC) is shifted and rounded as specified by SFMO and RND, respectively; the higher 24 bits of this value are stored in the addressed data memory location. Refer to the SFMO and RND instruction descriptions for more details on how these control bits affect the operand.

The MACC output value is checked for overflow twice, after it passes through both the Output Shifter and the Output Rounder. In the case that overflow is detected (MOV=1) and MOVM=1, depending upon the direction of overflow, the value is clipped to either 7FFFFFFFFFFFh or 800000000000h, rounded as specified by RND and bits 47–24 are stored in the addressed data memory location instead of the MACC output value. If MOVM=0, the MACC output value is not clipped.

Category

2B

Cycles

1 (Class 1)

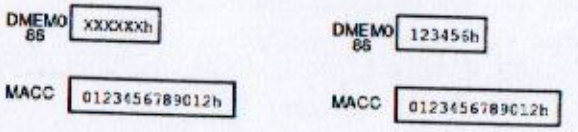
Repeatable

yes

Example

Given Conditions:BA0=0, DBP=0, MOVM=0, RND=0, SFMO=0.

```
SMHD D(86)
or
SMHD D* ;ID=86.
```



Syntax

Direct: **SMLD D(<dma>)**
Indirect: **SMLD {D*/D*+}**

Operands

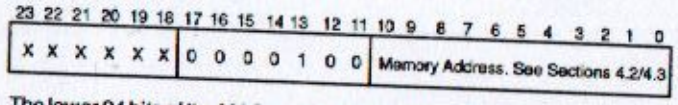
0 ≤ dma ≤ 255
0 ≤ dma ≤ 31 (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(MACC)_{23-0} \rightarrow dma_{23-0}$

Affected by MOVM, SFMO.
Affects MOV.

Encoding



Description

The lower 24 bits of the MAC accumulator (MACC) are stored in the addressed data memory location.

Although the MACC output is not affected by the Output Shifter and the Output Rounder, overflow detection is performed. The number of MACC bits checked for overflow is defined by the value of SFMO. Refer to the SFMO instruction for further details.

In the case that overflow is detected (MOV=1) and MOVM=1, depending upon the direction of overflow, the value is clipped to either 7FFFFFFFFFFFh or 800000000000h and bits 23–0 are stored in the addressed data memory location instead of the contents of the MACC. If MOVM=0, the MACC output value is not clipped. With this instruction, there is no rounding of a clipped result. See the SMOM instruction description for more details.

When using the SMHD and SMLD instruction in conjunction, it is recommended to set the SFMO control bit to zero.

Category

2B

Cycles

1 (Class 1)

Repeatable

yes

Example

Given Conditions:BA0=0, DBP=0, MOVM=0, SFMO=0.

```
SMLD D(86)
or
SMLD D* ;ID=86.
```

	Before Instruction:	After Instruction:
DMEM 86	000FA5h	789012h
MACC	0123456789012h	0123456789012h

Syntax

SMOM

Operands

None

Operation

(PC) + 1 → PC
1 → MOVM

Affects MOVM.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	1	0	0	0	0	0	1	X	X	X	X	X	X	X	X	X	X	X

Description

The SMOM instruction sets the MOVM (MACC overflow mode) control bit to logical one, which enables the overflow mode at the MAC accumulator (MACC) output. If overflow occurs (MOV=1), depending upon the direction of overflow, the overflowed value will be clipped to either the maximum positive (7F...Fh) or negative value (80...0h).

The direction of overflow is determined by the value of bit 51 of the MACC when overflow occurs (MOV is set to one). If bit 51 of the MACC is equal to zero, the direction of overflow is positive. If bit 51 of the MACC is equal to one, the direction of overflow is negative.

The clipping of the overflowed value occurs at the Overflow Limiter. After the overflowed value is replaced by either 7FFFFFFFFFFFh or 800000000000h, it will be rounded as specified by the RND control bit as shown below. Note that with the DOML, SLMH, SLML and SMLD instructions, no rounding is provided on a clipped value.

Rounding at Overflow Limiter

RND control bit	Positive Overflow Value ($MACC_{47-0}$)	Negative Overflow Value ($MACC_{47-0}$)
0	7FFF FFFF FFFF	8000 0000 0000
1	7FFF FFFF 0000	8000 0000 0000
2	7FFF FF00 0000	8000 0000 0000
3	7FFF F000 0000	8000 0000 0000
4	7FFF 0000 0000	8000 0000 0000

The overflow mode can be turned off by resetting MOVM to zero with the RMOM instruction.

Category

2N

Cycles

1 (Class 1)

Repeatable

no

Example

```
SMOM
```

Before Instruction:	After Instruction:
MOVM [0]	MOVM [1]

Syntax

Direct: **SRBD D(<dma>)**
Indirect: **SRBD {D*/D*+}**

Operands

$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(XRD)_{23-0} \rightarrow dma_{23-0}$

Affected by DBP.

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
X X X X X X	0 0 0 1 1 1 1 Memory Address. See Sections 4.2/4.3

Description

This instruction is available only for the TMS57002.

The contents of the external RAM interface read data buffer (XRD) are stored in the addressed data memory location. With the RDE instruction, one word of external memory data is loaded to XRD. If SRBD is executed before an RDE instruction is finished executing, the XRD buffer will contain the data from the execution of the previous RDE instruction. Refer to the RDE instruction for more details.

Category

2B

Cycles

1 (Class 1)

Repeatable

yes

Example

Given Conditions:BA0=0, DBP=0.

```
SRBD D(15)
or
SRBD D* ;ID=15.
```

Before Instruction:	After Instruction:
XRD [123456h]	XRD [123456h]
$DMEM0_{15}$ [XXXXXXh]	$DMEM0_{15}$ [123456h]

Syntax Indirect: **STD1 {D*/D*+}**

Operands None

Operation
(PC) + 1 → PC
(TREG) → $dma1_{23-0}$

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11	10 9 8 7 6 5 4 3 2 1 0
X X X X X X	0 0 0 1 1 0 1	Memory Address. See Sections 4.2/4.3

Description
The contents of TREG (temporary register) are stored in the addressed data memory location of DMEM1 (data memory bank1).

The setting of DBP does not affect the execution of this instruction.

This instruction is used for transferring a data value from DMEM0 to DMEM1 during calculations for linear phase FIR filter applications. The LD0T instruction transfers a DMEM0 value to TREG. Refer to Section 5.4 for more on linear phase FIR filter applications.

Category 2N

Cycles 1 (Class 1)

Repeatable yes

Example Given Conditions:BA1=159, ID=17.

```
STD1 D*
```

Before Instruction:		After Instruction:	
TREG	123400h	TREG	123400h
DMEM1 176	487500h	DMEM1 176	123400h

Syntax Direct: **SUB D(<dma>), A**
Indirect: **SUB {D*/D*+}, A**

Operands
0 ≤ dma ≤ 255
0 ≤ dma ≤ 31 (DMEM1 of TMS57002 only)

Operation
(PC) + 1 → PC
$(dma)_{23-0} - (AACC)_{31-0} \rightarrow AACC_{31-0}$

Affected by DBP, SFAI, SFAO.
Affects AOV.

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11	10 9 8 7 6 5 4 3 2 1 0
0 0 1 0 0 1	X X X X X X X	Memory Address. See Sections 4.2/4.3

Description
The data of the addressed data memory location are right–shifted as specified by SFAI, the output of the ALU accumulator (AACC) is shifted as specified by SFAO; the AACC value is subtracted from the data memory value and the result is stored in the AACC.

If the result overflows, AOV (AACC overflow flag) is set to one. AOV is cleared to zero at synchronization, reset or with the BV, INCD or RAOV instructions.

For a description of how the SFAI and SFAO control bits affect the operands, refer to the SFAI and SFAO instruction descriptions, respectively.

Category 1A

Cycles 1 (Class 2)

Repeatable yes

Example

Given Conditions:BA0=0, DBP=0, SFAI=0, SFAO=0.

```
SUB D(86), A
or
SUB D*, A ;ID=86.
```

Case 1:

Before Instruction:		After Instruction:	
DMEM0 86	000003h	DMEM0 86	000003h
AACC	00000156h	AACC	000001AAh
AOV	x	AOV	x

Case 2:

Before Instruction:		After Instruction:	
DMEM0 86	800001h	DMEM0 86	800001h
AACC	00000200h	AACC	7FFFFF00h
AOV	x	AOV	1

Syntax

Direct: **SUB C(<cma>), A**
Indirect: **SUB {C*/C*+}, A**

Operands

$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(cma)_{31-0} - (AACC)_{31-0} \rightarrow AACC_{31-0}$

Affected by CRM, SFAO.
Affects AOV.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	0	x	x	x	x	x	x	x	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed coefficient memory location are provided by CRM, the output of the ALU accumulator (AACC) is shifted as specified by SFAO; the AACC value is subtracted from the coefficient memory value and the result is stored in the AACC.

If the result overflows, AOV (AACC overflow flag) is set to one. AOV is cleared to zero at synchronization, reset or with the BV, INCD or RAOV instructions.

For a description of how the CRM and SFAO control bits affect the operands, refer to the CRM and SFAO instruction descriptions, respectively.

Category 1C

Cycles 1 (Class 2)

Repeatable yes

Example

Given Conditions:CRM=0, SFAO=0.

```
SUB C(125), A
or
SUB C*, A ;CA=125.
```

Case 1:

	Before Instruction:			After Instruction:		
CMEM 125	00000003h		CMEM 125	00000003h		
AACC	00000156h	AOV X	AACC	FFFFFEADh	AOV X	

Case 2:

	Before Instruction:			After Instruction:		
CMEM 125	00000003h		CMEM 125	00000003h		
AACC	80000003h	AOV X	AACC	80000000h	AOV 1	

Syntax

Direct: **SUB D(<dma>), M**
Indirect: **SUB {D*/D*+}, M**

Operands

$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(dma)_{23-0} - (MACC)_{47-16} \rightarrow AACC_{31-0}$

Affected by DBP, MOVM, RND, SFAI, SFMO.
Affects AOV, MOV.

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 0 1 0 1 1	X X X X X X X Memory Address. See Sections 4.2/4.3

Description

The data of the addressed data memory location are right–shifted as specified by SFAI, the output of the MAC accumulator (MACC) is shifted and rounded by SFMO and RND respectively; the MACC value is subtracted from the data memory value, and the result is stored in the ALU accumulator (AACC).

If the result overflows, AOV (AACC overflow flag) is set to one. AOV is cleared to zero at synchronization, reset or with the BV, INCD or RAOV instructions.

In the case of overflow when shifting or rounding the MACC output value, MOV (MACC overflow flag) is set to one. If MOVM=1, the MACC operand is clipped to 7FFFFFFFFFFh or 800000000000h and then rounded as specified by RND. Refer to the SMOM instruction for further details. If MOVM=0, the MACC output value is not clipped. MOV is cleared to zero at synchronization, reset or with the RMOV or INCD instructions.

For a description of how the DBP, MOVM, SFAI, SFMO and RND control bits affect the operands, refer to the LDPK, SMOM, SFAI, SFMO and RND instruction descriptions, respectively.

Category 1A

Cycles 1 (Class 2)

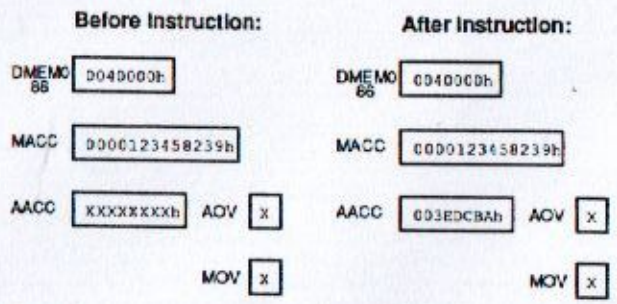
Repeatable yes

Example

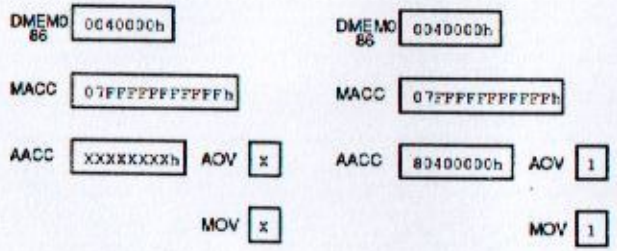
Given Conditions:BA0=0, DBP=0, SFAI=0, SFMO=0, RND=1.

```
SUB D(86), M
or
SUB D*, M ;ID=86.
```

Case 1:



Case 2:



Syntax

Direct: **SUB C(<cma>), M**
Indirect: **SUB {C*/C*+}, M**

Operands

$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(cma)_{31-0} - (MACC)_{47-16} \rightarrow AACC_{31-0}$

Affected by CRM, MOVM, RND, SFMO.
Affects AOV, MOV.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	0	x	x	x	x	x	x	x	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed coefficient memory location are provided by CRM, the output of the MAC accumulator (MACC) is shifted and rounded as specified by SFMO and RND, respectively; the MACC value is subtracted from the coefficient memory value, and the result is stored in the ALU accumulator (AACC).

If the result overflows, AOV (AACC overflow flag) is set to one. AOV is cleared to zero at synchronization, reset or with the BV, INCD or RAOV instructions.

In the case of overflow when shifting or rounding the MACC output value, MOV (MACC overflow flag) is set to one. If MOVM=1, the MACC operand is clipped to 7FFFFFFFFFFFh or 800000000000h and rounded as specified by RND. Refer to the SMOM instruction description for further details. If MOVM=0, the MACC output value is not clipped. MOV is cleared to zero at synchronization, reset or with the RMOV or INCD instructions.

For a description of how the CRM, MOVM, RND and SFMO control bits affect the operands, refer to the SCRM, SMOM, RND and SFMO instruction descriptions, respectively.

Category

1C

Cycles

1 (Class 2)

Repeatable

yes

Example

Given Conditions:CRM=0, SFMO=0, RND=1.

```
SUB C(125), M
or
SUB C*, M ;CA=125.
```

Case 1:

	Before Instruction:	After Instruction:
CMEM 125	00030000h	00030000h
MACC	0000123456789h	0000123456789h
AACC	XXXXXXXXh	0001DCBBh
AOV	x	x
MOV	x	x

Case 2:

	Before Instruction:	After Instruction:
CMEM 125	00030000h	80030000h
MACC	07FFFFFFFFFFFh	07FFFFFFFFFFFh
AACC	XXXXXXXXh	80030000h
AOV	x	1
MOV	x	1

Syntax

Direct: **SUB D(<dma>), {C*/C*+}**
SUB {D*/D*+}, C(<cma>)
Indirect: **SUB {D*/D*+}, {C*/C*+}**

Operands

$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)
$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(dma)_{23-0} - (cma)_{31-0} \rightarrow AACC_{31-0}$

Affected by CRM, DBP, SFAI.
Affects AOV.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	X	X	X	X	X	X	X	Memory Address. See Sections 4.2/4.3										

Description

The data of the addressed data memory location are right–shifted as specified by SFAI, the data of the addressed coefficient memory location are provided by CRM; the coefficient memory value is subtracted from the data memory value, and the result is stored in the ALU accumulator (AACC).

If the result overflows, AOV (AACC overflow flag) is set to one. AOV is cleared to zero at synchronization, reset or with the BV, INCD or RAOV instructions.

For a description of how the CRM and SFAI control bits affect the operands, refer to the CRM and SFAI instruction descriptions, respectively.

Category 1E

Cycles 1 (Class 2)

Repeatable yes

Example

Given Conditions:BA0=0, DBP=0, SFAI=0, CRM=0.

```
SUB D(86), C* ;CA=125.
or
SUB D*, C(125) ;ID=86.
or
SUB D*, C* ;CA=125, ID=86.
```

Case 1:

	Before Instruction:	After Instruction:
DMEM 86	000004h	000004h
CMEM 125	00000123h	00000123h
AACC	XXXXXXXXh	000002D0h
AOV	X	X

Case 2:

	Before Instruction:	After Instruction:
DMEM 86	000004h	000004h
CMEM 125	80000400h	80000400h
AACC	XXXXXXXXh	80000000h
AOV	X	1

Syntax

Direct: **WRE D(<dma>), {C*/C*+}**
WRE {D*/D*+}, C(<cma>)
Indirect: **WRE {D*/D*+}, {C*/C*+}**

Operands

$0 \le cma \le 255$, $0 \le dma \le 255$
$0 \le cma \le 31$ (DMEM1 of TMS57002 only)

Operation

$(PC) + 1 \rightarrow PC$
$(dma)_{23-0} \rightarrow XWR_{23-0}$
$(cma)_{18-0} \rightarrow XOA_{18-0}$
access address = XOA + XBA
$(XWR)_{23-0} \rightarrow$ external memory at access address

Affected by CRM, DBP, SEL, SRAM, WORD.

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	0	X	X	X	X	X	X	X	Memory address. See Section 4.2/4.3										

Description

This instruction is available only for the TMS57002.

The data of the addressed data memory location are loaded to the external RAM interface write data buffer (XWR). In parallel, the data of the addressed coefficient memory location are provided by CRM and the lower 19 bits of this value are loaded to the external RAM interface offset address register (XOA). The contents of the XWR buffer are then written to external memory.

The sum of XOA and XBA (external RAM interface base address register) are used in determining the access address into external memory. See Section 3.9.5 for more on external memory addressing.

The number of machine cycles required to write the contents of the XWR buffer to external memory is provided in Table 3–18 and Table 3–19. If an RDE, WRE or REF instruction is executed during this required number of machine cycles, the RDE, WRE or REF instruction will be ignored. The value of the SEL, SRAM and WORD control bits define the required number of machine cycles (see Table 3–15).

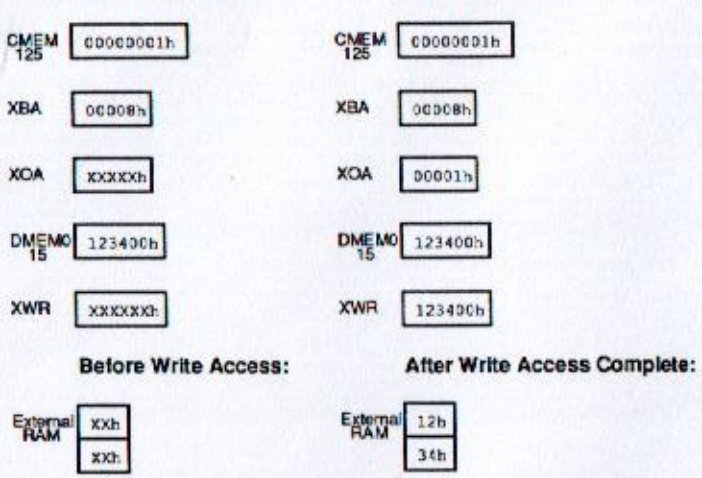
Category 1E

Cycles 1 (Class I)

Repeatable no

Example Given Conditions : BA0=0, CA=125, CRM=0, DBP=0, SEL=1, SRAM=0, WORD=0.

```
WRE D(15)
or
WRE D*, C(125) ;ID=15.
or
WRE D*, C* ;CA=125, ID=15.
```



Syntax Direct: **XOR D(<dma>), A**
Indirect: **XOR {D*/D*+}, A**

Operands $0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)

Operation $(PC) + 1 \rightarrow PC$
$(dma)_{23-0}$ or $(AACC)_{31-8} \rightarrow AACC_{31-8}$
$(AACC)_{7-0} \rightarrow AACC_{7-0}$

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0
0 1 1 0 1 0	X X X X X X X X	Memory address. See Section 4.2/4.3

Description The data of the addressed data memory location are right–shifted as specified by SFAI, the output of the ALU accumulator (AACC) is shifted as specified by SFAO; these two values are exclusive–ORed bitwise and the result is stored in the AACC. Note that the lower 8 bits of the shifted AACC output value are exclusive–ORed bitwise with all zeroes.

How the SFAI and SFAO control bits affect the operands is described in the SFAI and SFAO instruction descriptions, respectively.

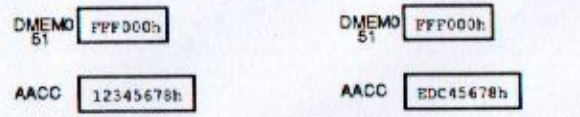
Category 1A

Cycles 1 (Class 2)

Repeatable no

Example Given Conditions:BA0=0, DBP=0, SFAI=0, SFAO=0.

```
XOR D(51), A
or
XOR D*, A ;ID=51.
```



Syntax

Direct: **XOR C(<cma>), A**
Indirect: **XOR {C*/C*+}, A**

Operands

$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(cma)_{31-0}$ or $(AACC)_{31-0} \rightarrow AACC_{31-0}$

Affected by CRM, SFAO.

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 1 1 0 1 1	X X X X X X X X X X X X X X X X X X
	Memory address. See Section 4.2/4.3

Description

The data of the addressed coefficient memory location are provided by CRM, the output of the ALU accumulator (AACC) is shifted as specified by SFAO; these two values are exclusive–ORed bitwise and the result is stored in the AACC.

How the CRM and SFAO control bits affect the operands is described in the CRM and SFAO instruction descriptions, respectively.

Category

1C

Cycles

1 (Class 2)

Repeatable

no

Example

Given Conditions:CRM=0, SFAO=0.

```
XOR C(187), A
or
XOR C*, A ;CA=187.
```

	Before Instruction:	After Instruction:
CMEM 187	0D0FFFFh	0030FFFFh
AACC	12345678h	1234A987h

Syntax

Direct: **XOR D(<dma>), {C*/C*+}**
XOR {D*/D*+}, C(<cma>)
Indirect: **XOR {D*/D*+}, {C*/C*+}**

Operands

$0 \le dma \le 255$
$0 \le dma \le 31$ (DMEM1 of TMS57002 only)
$0 \le cma \le 255$

Operation

$(PC) + 1 \rightarrow PC$
$(dma)_{23-0}$ XOR $(cma)_{31-8} \rightarrow AACC_{31-8}$
$(cma)_{7-0} \rightarrow AACC_{7-0}$

Affected by CRM, DBP, SFAI.

Encoding

23 22 21 20 19 18	17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 1 1 1 0 0	X X X X X X X X X X X X X X X X X X
	Memory address. See Section 4.2/4.3

Description

The data of the addressed data memory location are right–shifted as specified by SFAI, the data of the addressed coefficient memory location are provided by CRM; these two values are exclusive–ORed bitwise and the result is stored in the ALU accumulator (AACC). Note that the lower 8 bits of the coefficient memory value are exclusive ORed bitwise with all zeroes.

How the CRM, DBP and SFAI control bits affect the operands is described in the CRM, LDPK and SFAI instruction descriptions, respectively.

Category

1E

Cycles

1 (Class 2)

Repeatable

no

Example

Given Conditions:BA0=0, CRM=0, DBP=0, SFAI=0.

```
XOR D(86), C* ;CA=125.
or
XOR D*, C(125) ;ID=86.
or
XOR D*, C* ;CA=125, ID=86.
```

	Before Instruction:	After Instruction:
DMEM0 86	0D0FFFh	0D0FFFh
CMEM 125	12345678h	12345678h
AACC	XXXXXXXXh	123BA978h

Syntax ZACC

Operands None

Operation
(PC) + 1 → PC
0 → $AACC_{31-0}$

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Description The contents of the ALU accumulator (AACC) are zeroed.

Category 1N

Cycles 1 (Class 2)

Repeatable no

Example

```
ZACC
```

Before Instruction:		After Instruction:	
AACC	00001224h	AACC	00000000h

Syntax ZMAC

Operands None

Operation
(PC) + 1 → PC
0 → $MACC_{51-0}$

Encoding

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Description The contents of the MAC accumulator (MACC) are zeroed.

If ZMAC is executed following a multiplication instruction or MACC shift instruction, a pipeline conflict occurs. The ZMAC instruction takes precedence and the previous instruction is cancelled.

Category 1N

Cycles 1 (Class 1)

Repeatable no

Example

```
ZMAC
```

Before Instruction:		After Instruction:	
MACC	1234567890123h	MACC	000000000000h

Chapter 5

Hardware/Software Applications

Topic Page

5.1	Hardware Interfaces	5-2
5.2	Processor Initialization	5-8
5.3	Circular Buffer Implementation of Internal Data Memory	5-9
5.4	Linear Phase FIR Filters	5-11
5.5	Two Device Cascaded Linear Phase FIR Filters	5-20
5.6	Three Device Cascaded Linear Phase FIR Filters	5-25
5.7	Second Order IIR Filters	5-29

5.1 Hardware Interfaces

In many digital audio applications, it is advantageous to use digital signal processors as digital filters, surround processors, or various effectors.

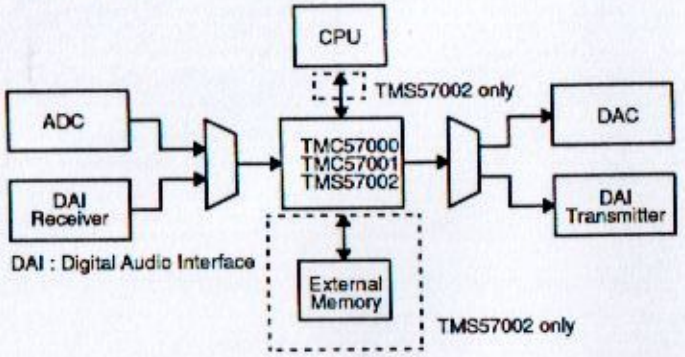
This hardware description describes methods for interfacing between peripherals and the 5700x.

The interfaces described are the following:

- Host interface
- Audio data interface
- External RAM interface
- TMS57014 interface

A sample hardware interface is shown in Figure 5–1.

Figure 5–1. Sample Hardware Interface



5.1.1 Host Interface

Via the host interface, data are transferred between the 5700x and the host controller. The types of data transfers available are listed below:

- Status register (ST0 and ST1) and program memory (PMEM) download
- Coefficient memory (CMEM) download
- Coefficient memory update
- CMEM/DMEM output to host controller (TMS57002 only)

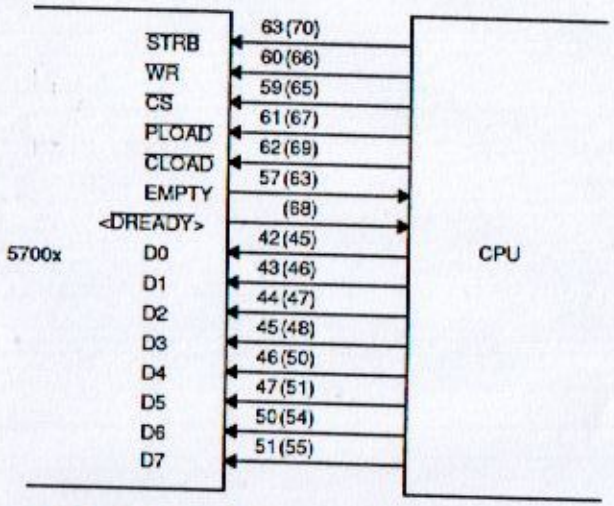
These are the key signals for this interface:

$\overline{\text{STRB}}$	Clock for latching data between D7 through D0 and HOST registers
$\overline{\text{WR}}$	Write signal (all devices except TMS57002)

R/$\overline{\text{W}}$	Read/Write signal (TMS57002 only)
$\overline{\text{CS}}$	$\overline{\text{STRB}}$ and $\overline{\text{WR}}$ (R/$\overline{\text{W}}$ for the TMS57002) enable signal
EMPTY	Indicates all UPDATE registers are empty
$\overline{\text{DREADY}}$	Handshaking signal for LPC/LPD instructions (TMS57002 only)
D7–0	8–bit input data bus (8–bit bidirectional data bus, TMS57002 only)

An example of the host interface is shown in Figure 5–2.

Figure 5–2. Host Interface



no–mark: 64–pin SDIP package
( ) : 80–pin QFP package
< > : TMS57002 only

Status register and program memory download, coefficient memory download, coefficient memory update, and CMEM/DMEM output to the host controller are described in Section 3.7.

5.1.2 Audio Interface

The 5700x series features a full–duplex on–chip serial port that provides direct communication with A/D converters, D/A converters and digital audio interfaces (DAI), especially DAI without any peripheral logic circuits. When interfacing with 5700x devices, some A/D converters and D/A converters need peripheral logic circuits added for data format conversion.

The 5700x serial I/O has been designed especially for the transmission of audio data. In one sampling period, two channels of data (left and right) are transmitted.

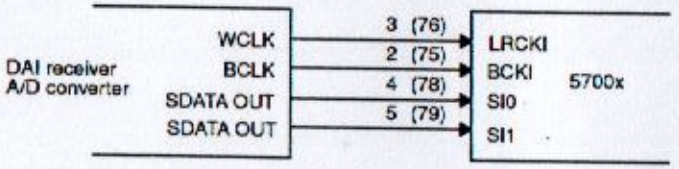
The transmit and receive sections of the serial port are implemented separately to allow independent transmit and receive operations. The LRCKI/LRCKO clock is the synchronization signal for one frame of input/output data. A frame is defined as one word of left and right channel data. The BCKI and BCKO clocks receive and transmit data, respectively.

These are the key signals for this interface:

LRCKI	Left/right channel framing synchronization signal for SIx
LRCKO	Left/right channel framing synchronization signal for SOx
BCKI	Receive data clock signal at SIx
BCKO	Transmit data clock signal at SOx
SI0/SI1	Received serial data signal at input port 0/1
SO0/SO1	Transmitted serial data signal at output port 0/1

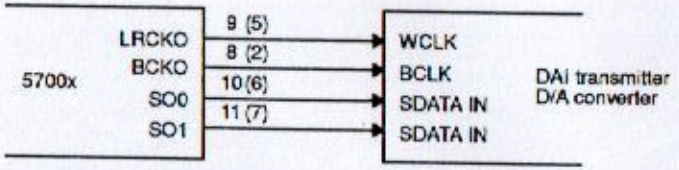
An example of an audio interface is shown in Figure 5–3. The DIRI, DIRO, FI, FO, PLRI, PLRO, PBCI, PBCO, SIM and SOM status register bits have to be set according to the data format of the target systems. See Section 3.8, Serial I/O Port.

Figure 5–3. Serial Data Input



no–mark: 64–pin SDIP package
( ) : 80–pin QFP package

Figure 5–4. Serial Data Output



no–mark: 64–pin SDIP package
( ) : 80–pin QFP package

5.1.3 External RAM Interface

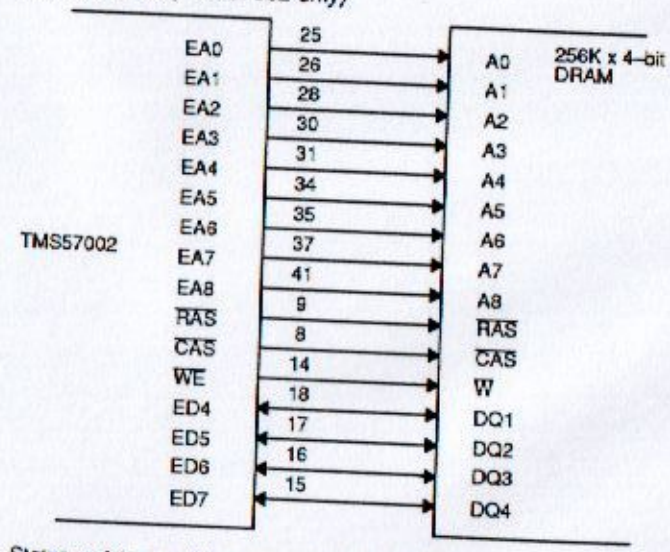
The architecture of the TMS57002 includes an external RAM interface. With the external RAM interface, the architecture of the TMC57000 and TMC57001 has been expanded to provide flexibility for the implementation of audio applications that require handling delayed data.

These are the key signals for this interface:

EA9–0	10–bit bidirectional address bus
ED7–0	8–bit bidirectional data bus
$\overline{RAS}$	Row address strobe signal
$\overline{CAS}$	Column address strobe signal
$\overline{WE}$	Write enable signal

An example of an external RAM interface with a 256k x 4–bit DRAM is shown in Figure 5–5. The conditions of the example are a 256k address space, a 4–bit I/O port width and DRAM with a 16–bit data width used as external RAM. The four control bits in ST0, M, SEL, SRAM and WORD have to be set to 1, 0, 0 and 0 respectively. In the case of a 4–bit I/O port width, the ED7 through ED4 are connected to the RAM. Thus the 4–bit data are transferred via ED7 through ED4.

Figure 5–5. External RAM Interface (TMS57002 only)



Status register conditions: M=1, SEL=0, SRAM=0 and WORD=0

5.1.4 TMS57014 Interface

The 5700x series can interface to the TMS57014, a 1–bit D/A converter for Hi–Fi digital audio. Refer to Figure 5–6 for the TMS57014 interface.

These are the key signals for this interface:

SYNC	Device synchronization signal
LRCKO	Left/right channel framing synchronization signal for SOx
BCKO	Transmit data clock signal at SOx

The TMS57014 supports an output data length of 16 or 18 bits. For 16–bit output data, BCKO/LRCKO ≥ 32 and for 18–bit output data, BCKO/LRCKO ≥ 36. The input data length and transmission order of the data bits (MSB or LSB first) are selected by setting DATLEN and INDIR respectively. The control bits of the 5700x and serial control data bits of the TMS57014 have to be set according to the following tables.

Table 5–1. 16–bit Data Transfer Mode

5700x control bits		TMS57014 serial control data bits	
Field	Value	Bit Name	Value
SOM	1 or 2	DATLEN	0
DIRO	0 (LSB first)	INDIR	1 (LSB first)
	1 (MSB first)		0 (MSB first)

Conditions: FO=0, PLRO=1, PBCO=0

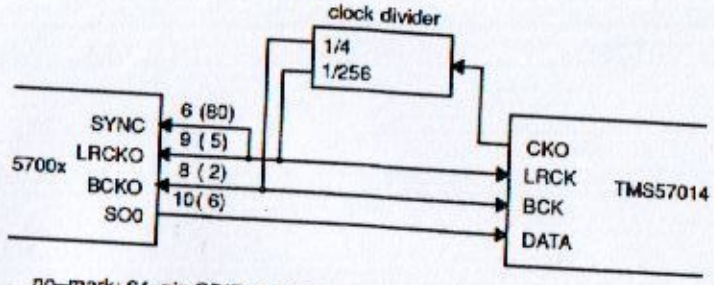
Table 5–2. 18–bit Data Transfer Mode

5700x control bits		TMS57014 serial control data bits	
Field	Value	Bit Name	Value
SOM	0	DATLEN	1
DIRO	0 (LSB first)	INDIR	1 (LSB first)

Conditions: FO=0, PLRO=1, PBCO=0

Regarding the other operations please refer to the TMS57014 data sheet.

Figure 5–6. TMS57014 Interface



no–mark: 64–pin SDIP package
( ) : 80–pin QFP package

5.2 Processor Initialization

Processor initialization includes resetting the device after power up and also initializing the contents of coefficient memory, program memory and the status registers.

A typical initialization sequence is listed below followed by a further explanation:

1) Power on. ;Apply +5 volts to V_{CC}.
2) Set $\overline{RS}$ signal low. ;Enter hardware reset state.
3) Input system clock. ;After 3 cycles, reset is complete.
4) Turn mute on. ;Set SO0/SO1 output to zero.
5) Set $\overline{RS}$ signal high. ;End reset cycle.
6) Set $\overline{PLOAD}$ signal low. ;Disable processor operation.
7) Download ST0, ST1, PMEM. ;Refer to Section 3.7.
8) Set $\overline{CLOAD}$ signal low. ;Refer to Section 3.7.
9) Download CMEM. ;Refer to Section 3.7.
10) Set $\overline{PLOAD}$ and $\overline{CLOAD}$ to high. ;Enable processor operation.
11) Turn mute off. ;Enable SO0/SO1 output.

At power up, the internal state of 5700x is undefined. The $\overline{RS}$ signal is needed to initialize internal hardware. The reset signal must be asserted low for at least three clock signals to guarantee a reset of the device. Refer to Section 3.6.5 for more details.

Before setting the $\overline{RS}$ signal high, set the $\overline{MUTE}$ signal low to disable noise data output.

After setting the $\overline{RS}$ signal high, set the $\overline{PLOAD}$ signal low to disable 5700x operation. Thus even if $\overline{RS}$ is set high, the device will not operate.

$\overline{PLOAD}$ and $\overline{CLOAD}$ are set high to enable the processor.

Set $\overline{MUTE}$ to high to enable serial output. It is recommended not to enable the $\overline{MUTE}$ signal until the contents of DMEM are stable.

5.3 Circular Buffer Implementation of Internal Data Memory

In this section, how DMEM0 has been implemented as a single circular buffer is discussed. With the TMC57000 and TMS57002(CAS=0), this discussion also applies to DMEM1. With the TMC57001 and TMS57002(CAS=1), DMEM1 is not implemented as a circular buffer as described in this section. This discussion also applies to external memory of the TMS57002. In addition, how to implement multiple circular buffers is also discussed.

For applications which require processing delayed audio data, a method is required that allows for easy storage and retrieval of delayed data. Therefore, DMEM0 has been implemented as a circular buffer. Because the entire DMEM0 area is used for the circular buffer implementation, the length of the DMEM0 circular buffer is 256.

Circular addressing has been employed for implementing circular buffers. With circular addressing, the data does not move but instead, by moving the base address, the data are effectively moved. The base address of DMEM0 is BA0. Before synchronization, BA0 points to the newest data sample in memory. At synchronization, BA0 is decremented. In the case that BA0=0, after decrementation BA0=255. The result of decrementing BA0 is that it now points to the memory location of the oldest data sample. The oldest data sample in memory can easily be replaced by the data sample just received. Thus through implementing circular addressing, without physically moving the data, the data are effectively moved in relation to the base address every time synchronization occurs.

At synchronization, in addition to BA0 being decremented, BA1 is incremented. BA1, the base address register for DMEM1, has been provided especially for implementing linear phase FIR filters. Refer to Section 3.3 and Section 5.4 for more on linear phase FIR filters. With the TMS57002, if INCS=1, the automatic decrementing and incrementing function of the base registers is disabled.

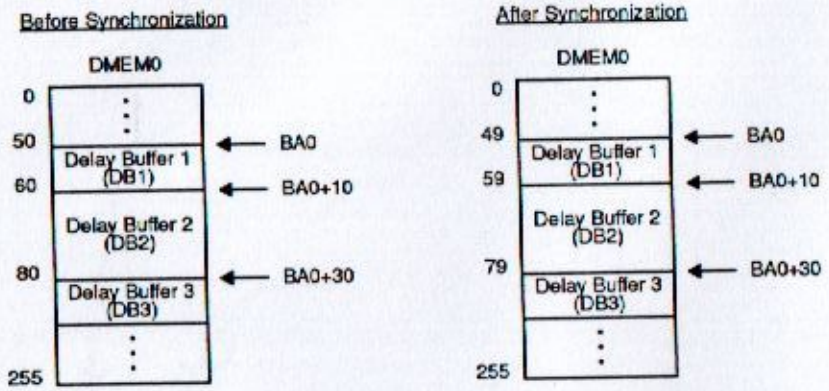
If DBP=0, DMEM0 is selected as the data memory bank. If DBP=1, DMEM1 is selected. When using the ADDS, AMAC AMPY, LD0t and STD1 instructions the DBP control bit is ignored.

To access a specific memory location, either direct or indirect addressing can be used. Assume you want to access a particular data sample K samples from the base address location. With direct addressing, simply access DMEM(K). The constant K will be loaded to IR_{7-0}, and then IR_{7-0} and BA0 are added to calculate the access address. With indirect addressing, the access address is calculated by adding ID (offset index) and BA0. ID (offset index) can be set to K using the LIRK instruction.

Within a circular buffer it is possible to create delay buffers of varying lengths. (A set of delayed data will be referred to as a delay buffer.) The starting location of a given delay buffer is defined as an offset address value from the base address value. This is not an absolute location but rather a relative location based upon the base address value. The location at which a new data sample is written to memory determines the starting location of a given delay buffer.

For example, as shown in Figure 5–7, DMEM0 contains three delay buffers, DB1, DB2 and DB3.

Figure 5–7. Internal Memory with Three Delay Buffers



The starting location of each of the data buffers is defined as BA0 + OFFSET. With indirect addressing, ID=OFFSET and with direct addressing, K=OFFSET. The start address of the delay buffers in Figure 5–7 are as follows:

```
DB1 start address = BA0
DB2 start address = BA0 + 10
DB3 start address = BA0 + 30
```

A new data sample for a given data buffer will be written at the location of the start address listed above. This start address location at which the new data sample is written also determines the size of the adjacent delay buffer. For example, in the case of the DB3 delay buffer, the start address of BA0 + 30 defines the buffer size of DB2 to be 20.

Although an FIR filter can be implemented with a single circular buffer, a second order IIR filter requires at least two delay buffers with a length of three. Refer to Section 5.4 and Section 5.7 for examples of FIR and IIR filter applications.

5.4 Linear Phase FIR Filters

There are many practical applications of linear phase FIR filters. With the 5700x, these applications can be implemented easily and with approximately half the processing time of conventional DSP. This is possible due to the separate multiplication and addition calculation units which allows for parallel processing of multiplication/accumulation and addition calculations, and also simultaneous data access of the two delayed data memory banks. Instructions specifically dedicated for linear phase FIR filter calculations (ADDS, AMACS, AMPY, MACS, LD0T and STD1) are provided.

The output of an N tap FIR filter can be given by the following equation:

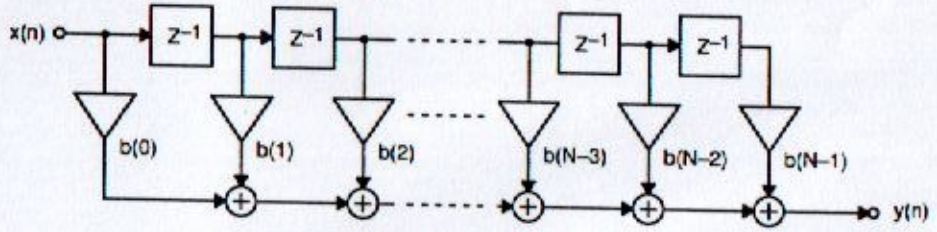
$$y(n) = \sum_{k=0}^{N-1} b(k)x(n-k)$$

where,

$$b(k) = b(N-1-k)$$

and can be represented by the following diagram:

Figure 5–8. N Tap Linear Phase FIR Filter



In the case of a linear phase FIR filter, the coefficients are symmetric. For a filter with an even number of taps,

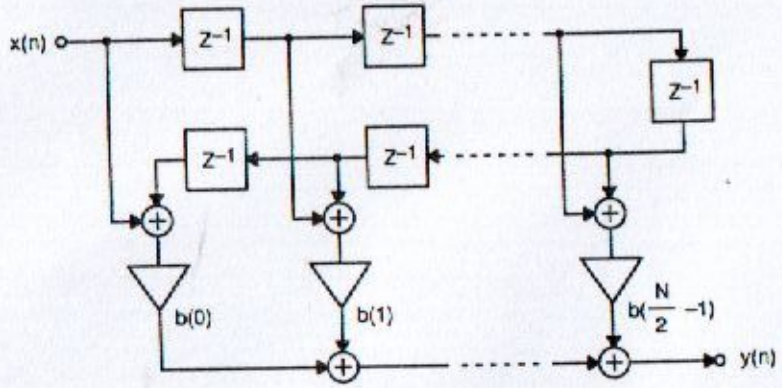
$$y(n) = \sum_{k=0}^{\frac{N}{2}-1} b(k)[x(n-k) + x(n-(N-1-k))]$$

and for a filter with an odd number of taps,

$$y(n) = \sum_{k=0}^{\frac{N-1}{2}-1} b(k)[x(n-k) + x(n-(N-1-k))] + b(\tfrac{N-1}{2})x[n-\tfrac{(N-1)}{2}]$$

A linear phase FIR filter with an even number of taps can be represented by Figure 5–9.

Figure 5–9. Linear Phase FIR Filter with Even Number of Taps



With linear phase FIR filters, the terms in the equation for y(n) have been reduced approximately in half. Specifically, for a filter with an even number of taps, the terms are reduced to (N/2)+1 and for a filter with an odd number of taps the number of terms are reduced to [(N–1)/2]+2. With the unique architecture of the 5700x, the execution cycles required to implement a linear phase FIR filter can be reduced by the same factor. The architecture which makes this possible is the following:

1) MAC and ALU: Two separate calculation units with parallel processing capability. The MAC is used for multiplication and accumulation.

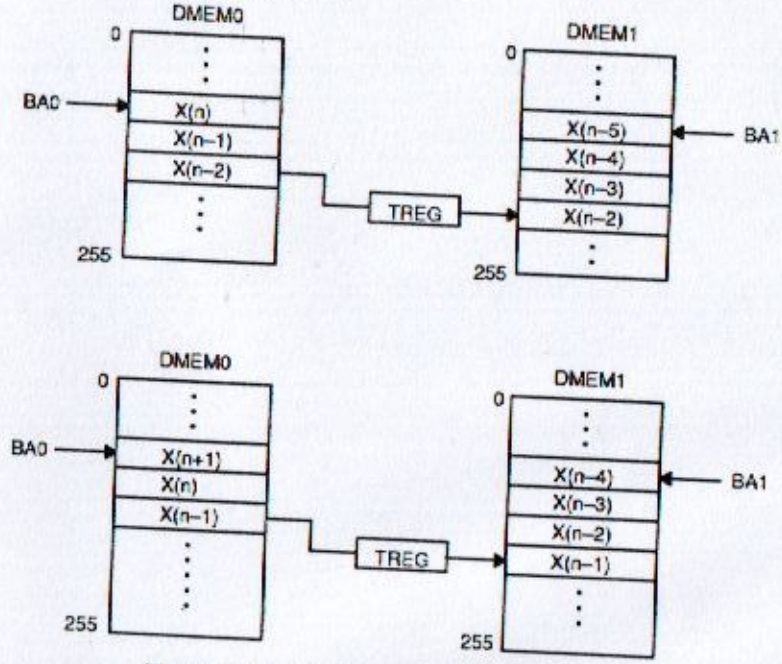
2) DMEM0 and DMEM1: Two data memory banks used for delayed data and an addressing scheme which allows for simultaneous access of both data memory banks.

In the case of a linear phase filter with an even number of taps, in DMEM0 the delayed data x(n–k) is stored, and in DMEM1 the delayed data x[n–(N–1–k)] is stored. By using the ADDS, AMPY, MACS, LD0T and STD1 instructions, x(n–k) and x[n–(N–1–k)] can be accessed simultaneously and the following executed in parallel:

- ☐ pre–addition of the addressed data samples
- ☐ multiplication of the previous pre–addition result and a coefficient

Both DMEM0 and DMEM1 are implemented as circular buffers of length 256 (with the TMS57002, the length of DMEM1 is 32). The base address register, BA0 points to the newest data sample, x(n). The base address register, BA1 points to the oldest data sample, x[n–(N–1)]. At synchronization, BA0 is decremented by one and BA1 is incremented by one. A new data sample is received and stored in DMEM0 at the address specified by BA0. The oldest data sample in DMEM0, x[n–((N/2) –1)] is transferred to DMEM1 via TREG using LD0T and STD1. This data transfer occurs before synchronization. Refer to Figure 5–10 for an illustration of the contents of DMEM0 and DMEM1 before and after synchronization for a 6 tap linear phase FIR filter.

Figure 5–10. DMEM Contents at Time n



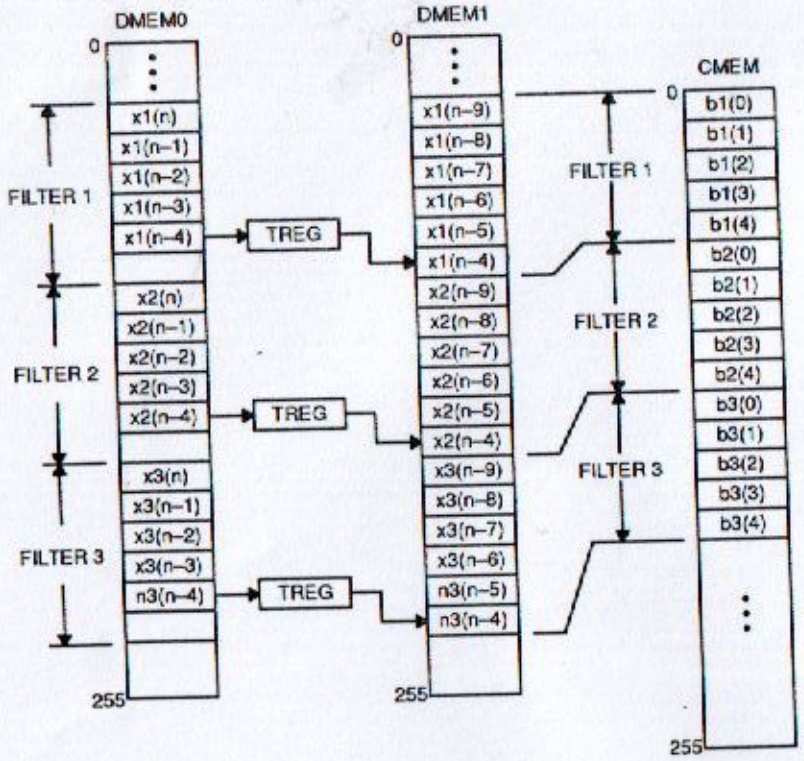
It is easy to implement cascaded linear phase FIR filters by creating a delayed buffer for each filter within data memory. Refer to Section 5.3 for more on the implementation of multiple delay buffers within one data memory bank. An example of implementing a three cascaded linear phase FIR filter with an even number of taps using the ADDS, AMPY, MAC, LD0T and STD1 instructions is provided in Example 5–1. Refer to Figure 5–11 as a reference to this example.

The algorithm for a linear phase FIR filter with an odd number of taps is slightly different. Refer to Example 5–2 for a sample program of a three cascaded linear phase FIR filter with an odd number of taps. See Figure 5–12 as a reference to this example.

Example 5–1. Triple Cascaded Linear Phase FIR Filter with 10 Taps

```
;;;
```

Figure 5–11. Memory Allocation for Triple Cascaded 10 Tap Linear Phase FIR Filter

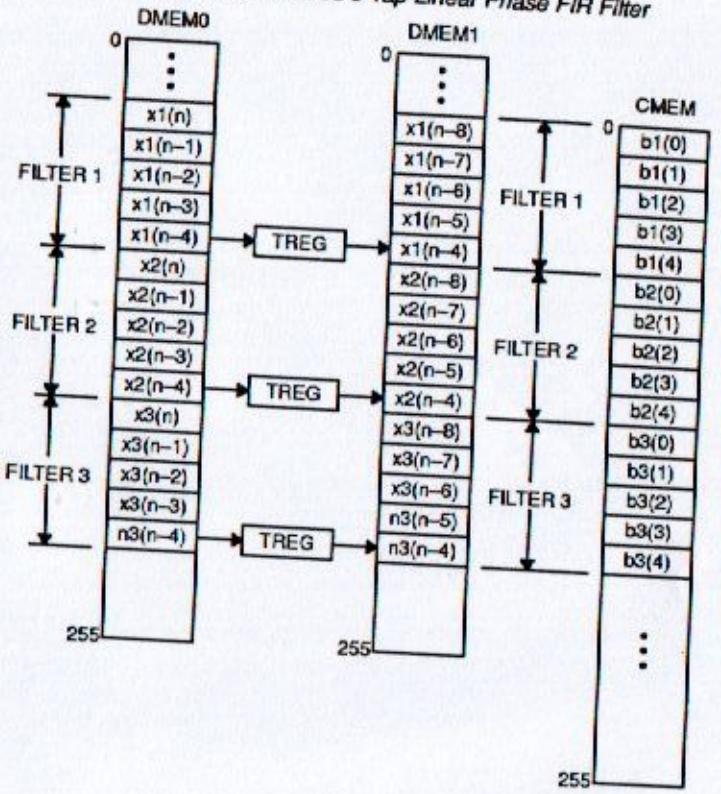


Example 5–2. Triple Cascaded Linear Phase FIR Filter with 9 Taps

```
;*************************************************************
;This routine implements three 9 tap linear phase FIR
;filters, FILTER 1, FILTER 2, and FILTER 3, which are
;cascaded together. The output of the three filters are
;represented as follows:
;
;                          3
; FILTER 1 output: y1(n)=SUM b1(k)[x1(n-k)+x1(n-8+k)]
;                          k=0                  +b1(4)x1(n-4)
;                 where x1(n)=new input
;
;                          3
; FILTER 2 output: y2(n)=SUM b2(k)[x2(n-k)+x2(n-8+k)]
;                          k=0                  +b2(4)x2(n-4)
;                 where x2(n)=y1(n)
;
;                          3
; FILTER 3 output: y3(n)=SUM b3(k)[x3(n-k)+x3(n-8+k)]
;                          k=0                  +b3(4)x3(n-4)
;                 where x3(n)=y2(n)
;
;Refer to Figure 5-12 for the memory allocation of this
;example.
;*************************************************************
;
;FILTER 1 calculation
;
ADDS D*+            /DIS 0    ; new input → DMEM0
                              ; pre-addition:
                              ; x1(n) + x1(n-8) → AACC
AMPY D*+, C*+, A    /SFMA 0   ; Initialize MACC Input
                              ; Shifter
                              ; b1(0) * AACC → MACC
                              ; x1(n-1) + x1(n-7) → AACC
AMAC D*+, C*+, A    /SFMO 0   ; Initialize MACC Output
                              ; Shifter
                              ; b1(1) * AACC + MACC → MACC
                              ; x1(n-2) + x1(n-6) → AACC
AMAC D*+, C*+, A    /RND 24   ; Initialize MACC Output
                              ; Rounder
                              ; b1(2) * AACC + MACC → MACC
                              ; x1(n-3) + x1(n-5) → AACC
MACS C*+, A                   ; b1(3) * AACC + MACC → MACC
MAC D*, C*+         /LDCT     ; x1(n-4) → TREG
NOP1                /STD1 D*+ ; TREG → DMEM1
```

```
;
;FILTER 2 calculation
;
ADDS D*+            /SMHD     ; result of FILTER 1 →DMEM0
                              ; pre-addition:
                              ; x2(n) + x2(n-8) → AACC
AMPY D*+, C*+, A              ; b2(0) * AACC → MACC
                              ; x2(n-1) + x2(n-7) → AACC
AMAC D*+, C*+, A              ; b2(1) * AACC + MACC → MACC
                              ; x2(n-2) + x2(n-6) → AACC
AMAC D*+, C*+, A              ; b2(2) * AACC + MACC → MACC
                              ; x2(n-3) + x2(n-5) → AACC
MACS C*+, A                   ; b2(3) * AACC + MACC → MACC
MAC D*, C*+         /LDOT     ; x2(n-4) → TREG
NOP1                /STD1 D*+ ; TREG → DMEM1
;
;FILTER 3 calculation
;
ADDS D*+            /SMHD     ; result of FILTER 2 →DMEM0
                              ; pre-addition:
                              ; x3(n) + x3(n-8) → AACC
AMPY D*+, C*+, A              ; b3(0) * AACC → MACC
                              ; x3(n-1) + x3(n-7) → AACC
AMAC D*+, C*+, A              ; b3(1) * AACC + MACC → MACC
                              ; x3(n-2) + x3(n-6) → AACC
AMAC D*+, C*+, A              ; b3(2) * AACC + MACC → MACC
                              ; x3(n-3) + x3(n-5) → AACC
                              ; Set MACC Output Rounder.
MACS C*+, A         /RND 16   ; b3(3) * AACC + MACC → MACC
MAC D*, C*+         /LDOT     ; x3(n-4) → TREG
NOP1                /STD1 D*+ ; TREG → DMEM1
NOP1                /DOMH 0   ; output of filter 3 result
;
```


Figure 5–12. Memory Allocation for Triple Cascaded 9 Tap Linear Phase FIR Filter



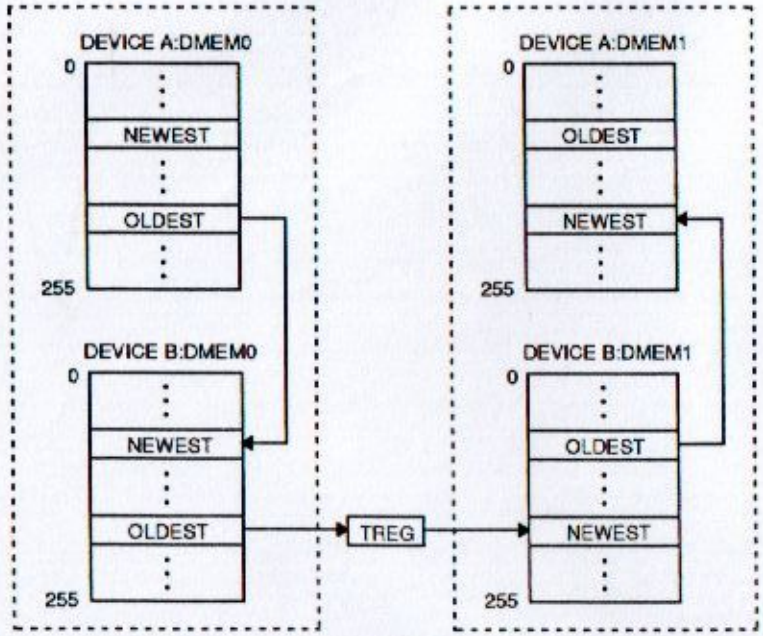
5.5 Two Device Cascaded Linear Phase FIR Filters

With the TMC57001 and TMS57002 (CAS=1) up to three devices can be cascaded when implementing linear phase FIR filters. This chapter explains how to implement a 500 taps linear phase FIR filter with two cascaded TMC57001 devices. A sample program is provided in Example 5–3.

The basic algorithm for implementing a cascaded linear phase FIR filter is the same as that for a single device linear phase FIR filter. The main point of difference between the two is that with cascaded devices, data must to transferred between the two devices. (The two devices will be referred to as DEVICE A and DEVICE B throughout this explanation).

The data memory transfer between devices is illustrated in Figure 5–13.

Figure 5–13. DMEM Data Transfer of 2 Cascaded Devices



This diagram reduces to the format for a one device linear phase FIR filter if the data banks enclosed in the perforated box are treated as one memory bank. Refer to Section 5.4 for more details on the implementation of a one device linear phase FIR filter. This material will not be treated in this section.

The hardware configuration of a two–device cascaded linear phase FIR filter application is illustrated in Figure 5–14. The serial I/O architecture, with two serial I/O ports, each supporting two 24–bit words of data transmission, makes it possible to cascade devices for linear phase FIR filter applications. As shown in Figure 5–14, at each serial I/O port, either one or two 24–bit data words are transferred.

The first half of the linear phase FIR filter calculation is calculated in DEVICE A. The result is then passed to DEVICE B where the second half of the calculation is continued. The 48–bit result from DEVICE A is transferred to DEVICE B as two 24–bit words, High MACC and Low MACC as illustrated in Figure 5–14. After DEVICE B's calculation is complete, the result is outputted.

When a new data sample is received at synchronization, it is stored in DMEM0 of DEVICE A. The remaining data memory locations must be updated. As illustrated in Figure 5–13, the oldest data sample of DEVICE A, DMEM0 becomes the newest data sample of DEVICE B, DMEM0. Similarly, the oldest data sample of DEVICE B, DMEM1 becomes the newest data sample of DEVICE A, DMEM1. The transfer of data between devices is illustrated in detail in Figure 5–15.

Figure 5–14. Hardware Configuration of 2 Cascaded Devices

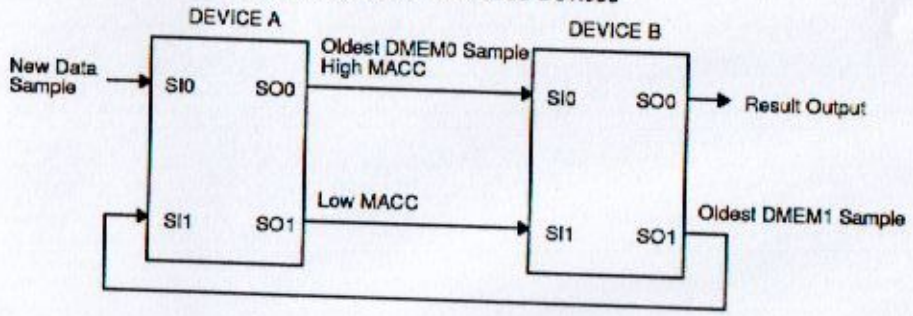
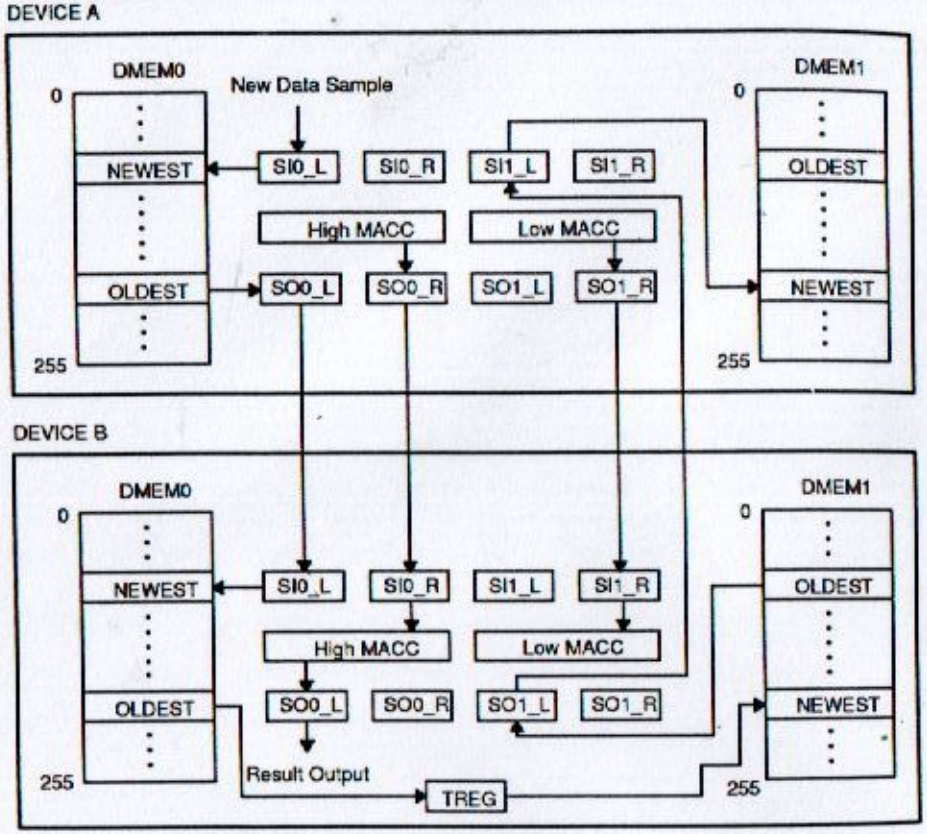
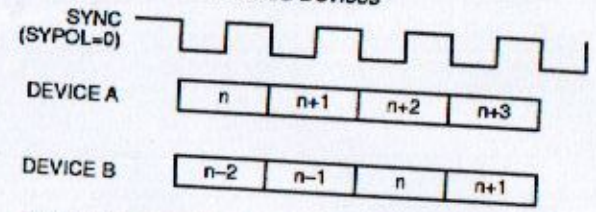


Figure 5–15. Register Contents of 2 Cascaded Devices at Time n



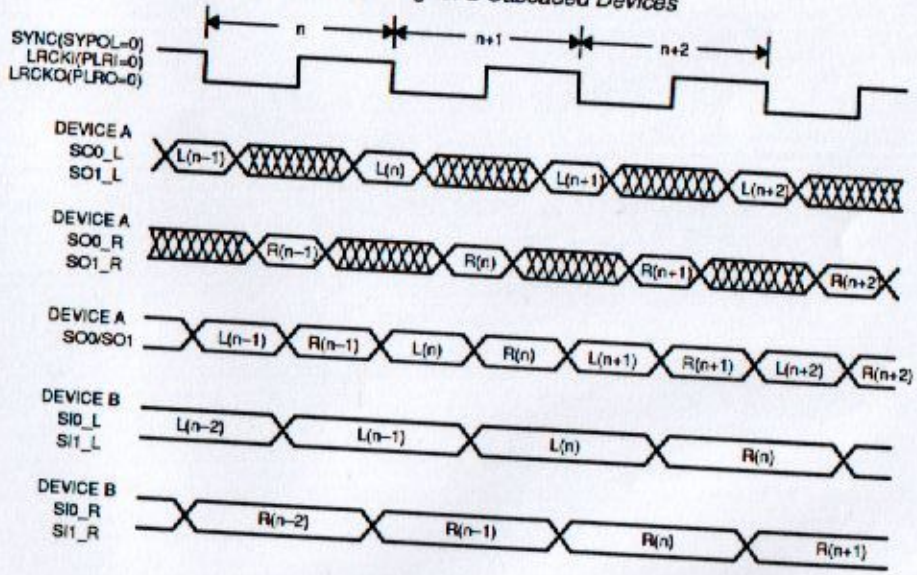
One linear phase FIR calculation requires two cycles as shown by the pipelined operation of DEVICE A and DEVICE B in Figure 5–16.

Figure 5–16. Pipeline Operation of 2 Cascaded Devices



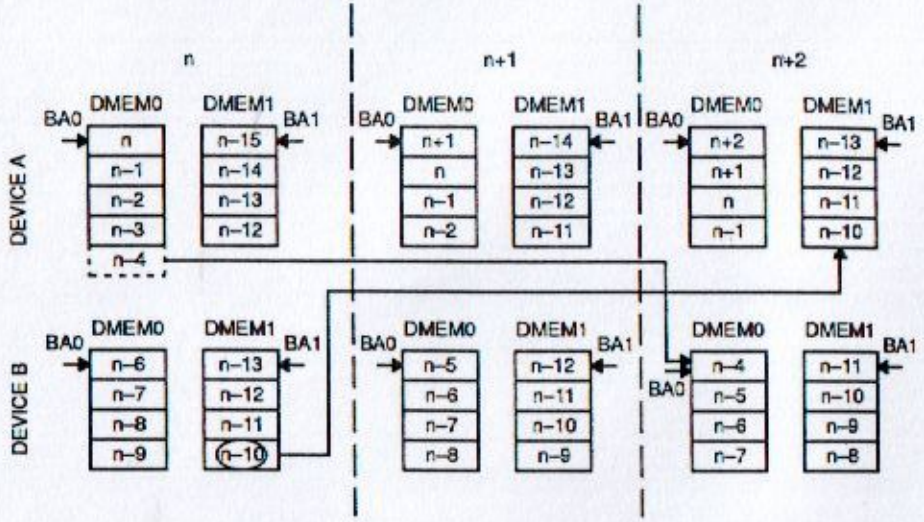
Two cycles are required because of the time required to transfer data between devices. This timing is illustrated in Figure 5–17. Note that the DOS instruction must be executed in PMEM no later than at address FAh.

Figure 5–17. Serial Port Register Timing for 2 Cascaded Devices



The contents of data memory for DEVICE A and DEVICE B at time n, n+1 and n+2 for a 16 tap linear phase FIR filter are illustrated in Figure 5–18. At time n in DEVICE A, the first half of the linear phase filter calculation is calculated, and at time n+2 in DEVICE B, the remaining half of the calculation is completed.

Figure 5–18. DMEM Contents of 2 Cascaded Devices at Time n, n+1 and n+2



For one linear phase FIR filter calculation at time n, two data memory transfers between devices are required. Firstly, in DEVICE A at time n, the fourth data sample from the DMEM0's base address register (BA0) must be outputted to DEVICE B in order for the sample to be available at time n+2. Secondly, at time n–2, the third data sample from DMEM1's base address register (BA1) must be outputted to DEVICE A in order for the sample to be available at time n.

Refer to Figure 5–17 for the serial port timing for the transfer of data between 5700x devices.

Example 5–3. Linear Phase FIR Filter for 2 Cascaded Devices

```
;;;
```

5.6 Three Device Cascaded Linear Phase FIR Filters

With the TMC57001 and TMS57002 (CAS=1), up to three 5700x devices can be cascaded when implementing linear phase FIR filters. When three TMC57001 devices are cascaded, a linear phase FIR filter with up to 1500 taps can be implemented. The basic algorithm for implementing a linear phase FIR filter using three cascaded devices is discussed in this section. The implementation of three cascaded devices is similar to the implementation of two cascaded devices. Please refer to Section 5.5, Two Device Cascaded Linear Phase FIR Filters, as a reference.

The serial I/O architecture, with two serial I/O ports, each supporting two 24–bit words of data transmission, makes it possible to cascade up to three devices for linear phase FIR filter applications. The hardware configuration of a three–device cascaded linear phase FIR filter application is illustrated in Figure 5–19. As shown in Figure 5–19, at each serial I/O port, either one or two 24–bit data are transferred.

Similar to the application for two–device cascaded linear phase filters, the result from DEVICE A must be transferred to DEVICE B, and the result from DEVICE B must be transferred to DEVICEC. The result is transferred as two 24–bit words, High MACC and Low MACC as illustrated in Figure 5–19.

When implementing a linear phase FIR filter with three devices, data from DMEM1 of DEVICE C can not be loaded directly to DMEM1 of DEVICE B as conceptually illustrated in Figure 5–20. The data from DEVICE C is first passed through DEVICE A before being stored in DEVICE B. This is illustrated in Figure 5–19.

The output of DEVICE C at SO1, DEVICE C: DMEM1 is the oldest data sample in DEVICE C. This value is inputted to SI1 of DEVICE A. This value is then output at SO1 of DEVICE A as noted by DEVICE C: DMEM1. This value is then loaded to SI1 of DEVICE B and stored in DEVICE B.

Similarly, data from DMEM1 of DEVICE B must be passed through DEVICE C before being loaded to DEVICE A. As illustrated in Figure 5–19, the output of DEVICE B at SO1, DEVICE B: DMEM1 is the oldest data sample in DEVICE B. This value is input to SI1 of DEVICE C. This value is then outputted at SO1 of DEVICE C as noted by DEVICE B: DMEM1. This value is then loaded to SI1 of DEVICE A and stored in DEVICE A.

Figure 5–19. Hardware Configuration of 3 Cascaded Devices

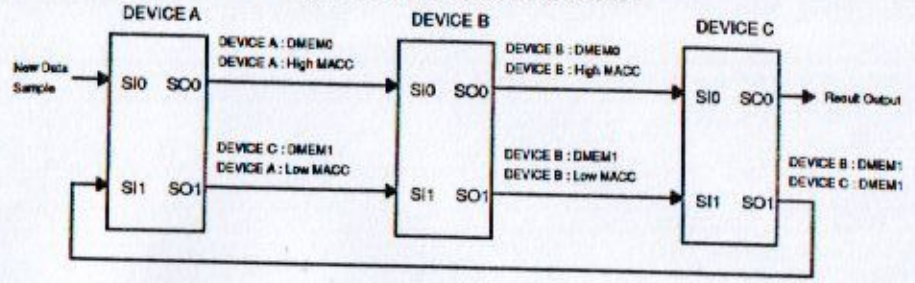
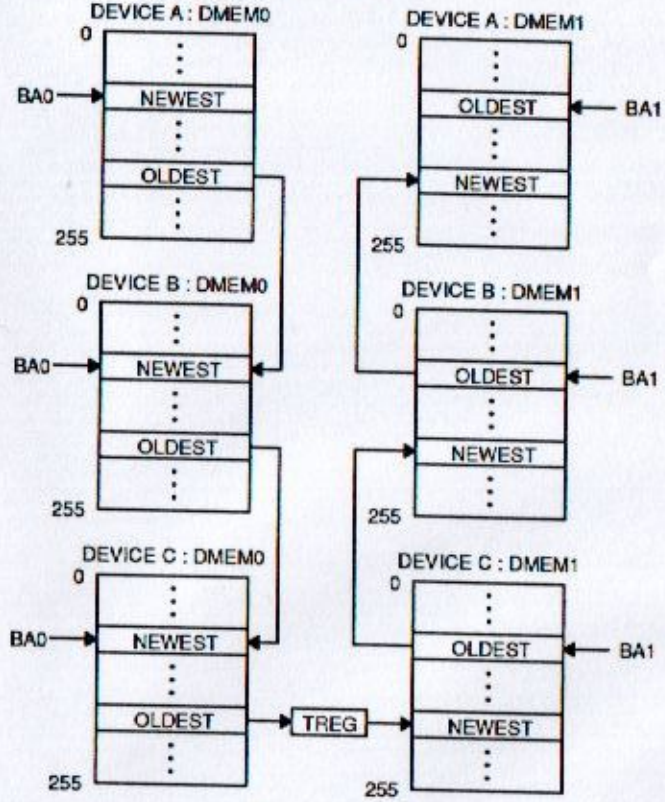


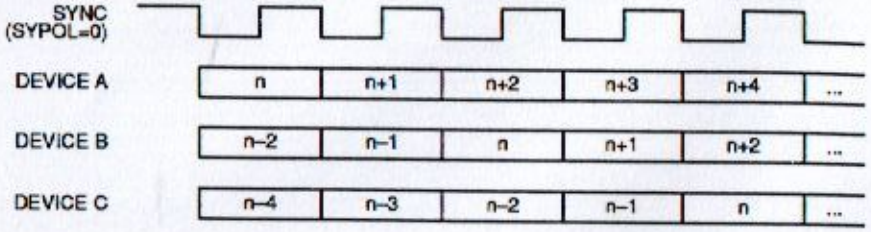
Figure 5–20. DMEM Data Transfer of 3 Cascaded Devices



As a result of the serial I/O timing, two SYNC cycles are required to transfer data between two devices. Therefore, in the case of three–device cascaded architecture, it takes five SYNC cycles to complete one linear phase FIR calculation as shown by the pipeline operation in Figure 5–21.

When writing code for this application, be sure to take this data transfer timing into consideration. Refer to Example 5–3 on page 524-25 for sample code of a linear phase FIR filter with two cascaded devices.

Figure 5–21. Pipeline Operation of 3 Cascaded Devices



5.7 Second Order IIR Filters

Second Order Infinite Impulse Response(IIR) Filters are widely used in digital signal processing applications. The 5700x offers an IIR solution with a high degree of precision (24–bit DMEM by 32–bit CMEM). An implementation for a long word solution (40–bit DMEM by 32–bit CMEM) is also presented here.

The output of a second order IIR filter can be given by the following equation:

$$Y(n) = \sum_{k=1}^{2} a(k)y(n-k) + \sum_{k=0}^{2} b(k)x(n-k)$$

where

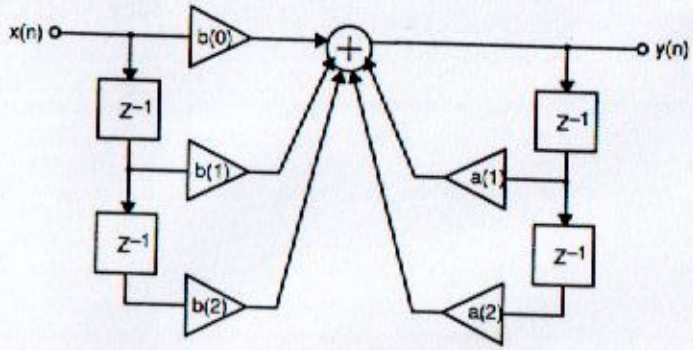
$a(k) = constant$

and

$b(k) = constant$

and can be represented by the following diagram:

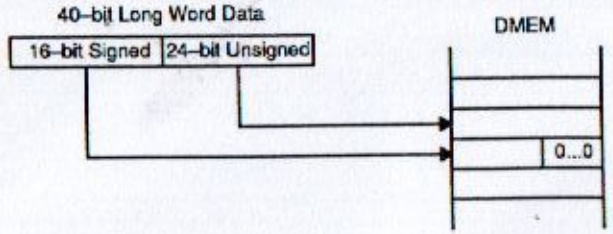
Figure 5–22. Second Order IIR Filter



In Example 5–4, a sample program for the implementation of a single precision (24–bit DMEM word by 32–bit CMEM word) second order IIR Filter is listed. See Figure 5–24 as a reference to this example.

For cases in which greater precision is necessary, a sample program for the implementation of a long word second order IIR filter (40–bit DMEM word by 32–bit CMEM word) is provided in Example 5–5. See Figure 5–25 and Figure 5–26 as references to this example. The data format for the 40–bit DMEM word is shown in Figure 5–23.

Figure 5–23. IIR Filter 40–bit Long Word Data Format



To multiply the 40–bit DMEM word by the 32–bit CMEM word, first unsigned multiplication is performed on the lower 24 bits of the long DMEM word and 32–bit CMEM word. The MACU and MPYU instructions are available especially for this application. This result is then shifted 16 bits to the right, and signed multiplication is performed on this shifted value and the higher 16 bits of the long DMEM word.

Example 5–4. Triple Cascaded Second Order IIR Filter (24–bit DMEM by 32–bit CMEM)

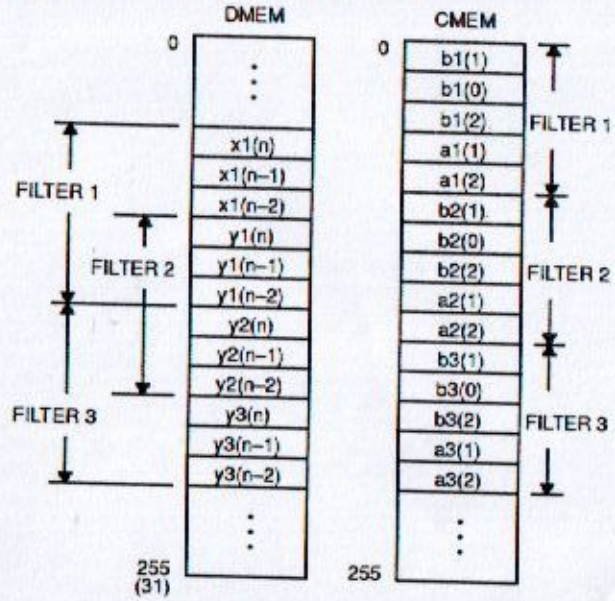
```
;;;
```

```
;
;FILTER 2 calculation
;
MPY D(4), C*+                   ; Start FILTER 2 calculation
MAC D(3), C*+       /SMHD       ; Store result of FILTER 1
MAC D(5), C*+
MAC D(7), C*+
MAC D(8), C*+                   ; End FILTER 2 calculation

;
;FILTER 3 calculation
;
MPY D(7), C*+                   ; Start FILTER 3 calculation
MAC D(6), C*+       /SMHD       ; Store result of FILTER 2
MAC D(8), C*+
MAC D(10), C*+
MAC D(11), C*+                  ; End FILTER 3 calculation
;

;
;Output Result of FILTER 3
;
NOP1
NOP1                SMHD D(9) ; Store result of FILTER 3
NOP1                RND 16    ; Set Output Rounder
NOP1                DOMH 0    ; Output FILTER 3 result
```


Figure 5–24. Memory Allocation for Single Precision IIR Filter

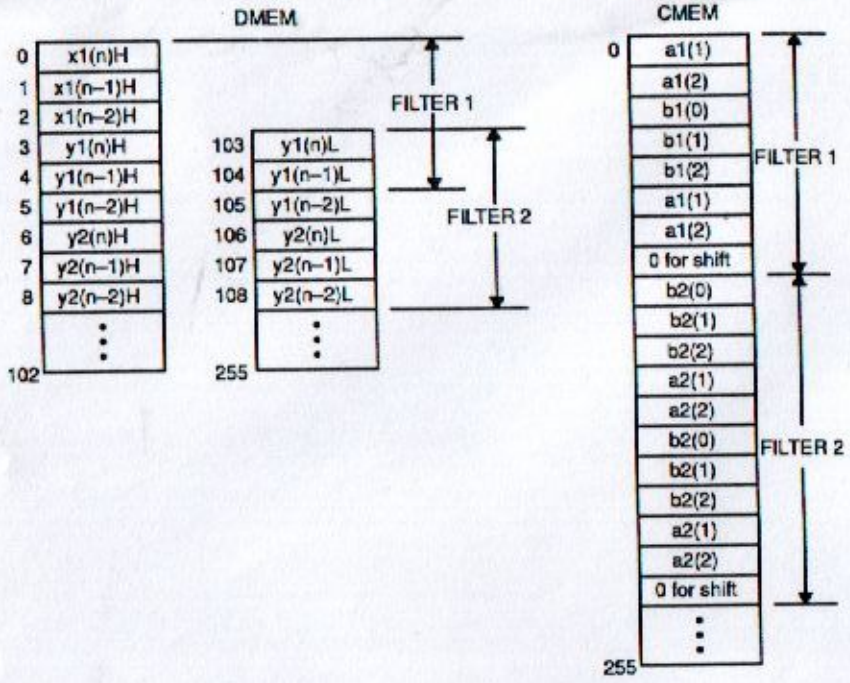


(31) : DMEM1 of TMS57002 only

Example 5–5. Double Cascaded Second Order IIR Filter (40–bit DMEM by 32–bit CMEM)

```
;;;
```

Figure 5–26. Memory Allocation for Long Word IIR Filter (40–bit DMEM by 32–bit CMEM)



Appendix A

Electrical Specifications

Topic

		Page
A.1	TMC57000/TMC57001 Electrical Specifications	A-2
A.2	TMS57002 Electrical Specifications	A-12